ioProgrammo Anno XI - N°1 (110)

ANTEPRIMA MS EXPRESSION TOOLS
I NUOVI STRUMENTI PER LA CREAZIONE DELLE INTERFACCE GRAFICHE, ANCHE 3D

Rivista + "Le grandi guide di ioProgrammo" n°8 a € 12,90 in più
VERSIONE PLUS RIVISTA+LIBRO+CD €9,90
VERSIONE STANDARD RIVISTA+CD €6,90

PER ESPERTI E PRINCIPIANTI
Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile • GENNAIO 2007 • ANNO XI, N.1 (110)

APPLICAZIONI PRONTE PER VISTA

IL SISTEMA DEL FUTURO È ALLE PORTE! FATTI TROVARE PREPARATO

Usa il .NET framework 3.0 e rendi subito disponibile il tuo codice per la nuova piattaforma

Teoria e tecnica
Cosa cambia nel linguaggio e nelle librerie. Le cose da sapere per programmare meglio

La migrazione
Passo dopo passo le cose da fare per portare il tuo software al nuovo sistema senza problemi

La pratica
Facciamo convivere le vecchie Windows Form con il nuovo XAML in modo semplice

NEL CD
CHATTA CON AJAX

Metti in comunicazione gli utenti del tuo sito, senza installare un server e fornendogli un'applicazione web che ha tutte le comodità di un software standalone

.NET FRAMEWORK

CREIAMO REPORT IN FORMATO RTF

Ecco come creare documenti direttamente comprensibili da Microsoft Word

NOVITÀ

IRONPYTHON: ARRIVA IL PITONE SECONDO .NET

Scopri, Il linguaggio che fa muovere Google ma ottimizzato per le tecnologie Microsoft

SISTEMA

UN CALENDARIO UNIVERSALE JAVA

Ecco come fanno Outlook e gli altri a scambiarsi gli appuntamenti

APPLICAZIONI MULTILINGUA .NET

Crea software pronto per il mondo senza dover riscrivere il codice

GESTIRE BENE LA MEMORIA C++

le tecniche da utilizzare affinché il tuo programma non vada fuori controllo

METTIAMO UN PDF DENTRO L'IPOD JAVA

Convertiamolo in un formato comprensibile dal famoso lettore

SOFTWARE NEL BROWSER .NET

Sfrutta AJAX per creare interfacce completamente indipendenti dal Web

WEB

UN MODULO DI REWRITE PER IIS .NET

Cloniamo alcune funzioni tipiche dell'htaccess di Apache usando gli HttpHandler

COMPRENDERE LE REGEX JAVASCRIPT

Per effettuare nel testo le ricerche che non si possono fare in altro modo

CORSI

ECLIPSE E LE REVISIONI JAVA

Versioni sempre sotto controllo con gli automatismi del tuo IDE preferito

SOLUZIONI Algoritmi evolutivi, ovvero sviluppare software che autoapprende dalle proprie esperienze

EDIZIONI MASTER
www.edmaster.it
60110
9 771128 594009

APPLICAZIONI PRONTE PER VISTA • CHATTA CON AJAX • REPORT IN FORMATO RTF • IRONPYTHON, IL PITONE SECONDO .NET • COMPRENDERE LE REGEX
EDIZIONI MASTER

Anno XI - N.ro 01 (110) - Gennaio 2007 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Vice Publisher: Paolo Soldan
Redazione: Fabio Farnesi
Collaboratori: R. Allegra, L. Buono, A. Galeazzi, F. Grimaldi, F. Smelzo, A. Pelleriti, M. Locuratolo, L. Corias
Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Art Director: Paolo Cristiano
Responsabile grafico di progetto Salvatore Vuono
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Francesco Cospite

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti
Direttore Generale: Massimo Rizzo

ABBONAMENTO E ARRETRATI

ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90 Offerte valide fino al 31/03/07 costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito Visa, Cartasì, o Eurocard/Mastercard (inviando la Vs. autorizzazione, il numero di carta di credito, la data di scadenza, l'intestatario della carta e il codice CVV2, cioè le ultime 3 cifre del codice numerico riportato sul retro della carta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o BCC MEDIOCRATI S.C.AR.L. c/c 0 000 000 12000 ABI 07062 CAB 80880 CIN P (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Servizio Abbonati:
☎ tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Assistenza tecnica: ioprogrammo@edmaster.it
Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Dicembre 2006

100 Cellulari, 100 Computer, 100 Fotocamere e Videocamere, 100 Palmari e GPS, 100 Stampanti e Consumabili, 100 TV LCD e Plasma, Audio/Video/Foto Bild Italia, A-Team, Calcio & Scommesse, Colombo, Computer Bild Italia, Computer Games Gold, Digital Japan Magazine, Digital Music, Distretto di Polizia in DVD, DVD Magazine, DVD Magazine Films, Family DVD Games, Filmteca in DVD, Go!OnLine Internet Magazine, Home Entertainment, Horror Mania, I Corsi di Win Magazine, I DVD di Win Magazine I DVD de La Mia Barca, I Fantastici CD-ROM, I Film di Idea Web, I Filmissimi in DVD, I Grandi Giochi per Pc, I Libri di Quale Computer, I Mitici all'Italiana, Idea Web, InDVD, IoProgrammo, I Tecnoplus di Win Magazine, Japan Cartoon, La mia Barca, La mia Videoteca, Le Femme Fatale del Cinema, Le Grandi Guide di Io Programmo, Linux Magazine, Magnum PI, Miami Vice in DVD, Nightmare, Office Magazine, Play Generation, Play Generation Games, Popeye, PC Junior, PC VideoGuide, Quale Computer, Softline Software World, Sport Life, Supercar in DVD, Star in DVD, Video Film Collection, Win Junior, Win Magazine Giochi, Win Magazine, Le Collection.

Questo mese su ioProgrammo

UN NODO CRUCIALE

Capita di tanto in tanto in informatica, un momento in cui improvvisamente tutto sembra cambiare radicalmente. E' stato così per il passaggio da windows 3.1 a Windows 95 e poi a Windows 98. E' stato così per il passaggio di Mac al nuovo kernel basato su BSD, ed è stato così quando sono nati i primi linguaggi di scripting per il Web. Così oggi ci siamo di nuovo. E' il momento di Windows Vista. Questa volta il carico di novità è importante e lo è ancora di più per chi programma con le tecnologie Microsoft. Il nuovo sistema operativo presenta un'enorme serie di novità. E d'altra parte il .NET framework 3.0 segue di pari passo le nuove caratteristiche associate al sistema per cui è stato pensato: Vista. E' anche vero che i layer che vengono montati sopra il nuovo framework migliorano ed estendono la precedente versione senza per questo stravolgerla, ma è anche vero che le novità sono talmente tante e dense di innovazioni tecnologica che è impossibile non annoverare il momento del rilascio di Windows Vista con un momento di svolta cruciale per l'evoluzione dell'informatica in generale. Tanto più che insieme a questa importante novità se ne presentano altre che rendono questo inizio di nuovo anno un momento eccezionale. Java 6 è alle porte, la diffusione di Linux assume proporzioni decisamente degne di nota. XGL e AIXGL sono perfettamente paragonabili alle funzioni 3D presenti in Vista. Si sta studiando alla nuova versione di PHP. Insomma tutto sembra muoversi di concerto per far si che questo 2007 venga ricordato come l'anno in cui il mercato ha ricominciato a muoversi. Per noi sviluppatori è tempo di migrazione, di portare le nostre applicazioni su piattaforme tecnologicamente innovative, è il momento di consigliare ai nostri clienti strade nuove e più produttive. Se vogliamo cogliere questa occasione dobbiamo essere pronti a stare al passo con i tempi, ma d'altra parte chi meglio di noi può farlo?

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre \soft\codice\ e \soft\tools\) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

Applicazioni pronte per VISTA

IL SISTEMA DEL FUTURO È ALLE PORTE! FATTI TROVARE PREPARATO

Usa il .NET framework 3.0 e rendi subito disponibile il tuo codice per la nuova piattaforma

- ✔ **Teoria e tecnica: cosa cambia nel linguaggio e nelle librerie. Le cose da sapere per programmare meglio**
- ✔ **La migrazione: Passo dopo passo le cose da fare per portare il tuo software al nuovo sistema senza problemi**
- ✔ **Pratica: Facciamo convivere le vecchie Windows Form con il nuovo XAML in modo semplice**

pag. 16

CHATTA CON AJAX

Metti in comunicazione gli utenti del tuo sito, senza installare un server e fornendogli un'applicazione Web che ha tutte le comodità di un software standalone

pag. 45

IOPROGRAMMO WEB

Gli strumenti di sviluppo del futuro . . . . **pag. 30**
Expression: é questo il nome che Microsoft ha voluto dare alla sua nuova suite per la creazione di interfacce grafiche e contenuti multimediali. La novità è che questa volta tutto è pensato per aiutare gli sviluppatori

Una mailing list gestita sul Web **pag.36**
Abbiamo realizzato un servizio che gestisce una lista di distribuzione della posta senza aver installato nessun software lato server ma solo delle pagine Web. Vediamo ora come creare il clienty per la consultazione

IIS come Aphace con l'Url Rewriting . . . . **pag. 52**
Si tratta di una delle funzionalità più amate dagli utilizzatori del noto Web server opensource. Consente di eliminare i parametri lunghissimi che solitamente caratterizzano una Queri_string. Creiamone uno per IIS

Javascript e le regular expression . . . . **pag. 58**
In questo articolo apprenderete cosa sono le espressioni regolari e come utilizzarle in Javascript. Vedremo come sarà possibile validare, lato client, e-mail e quant'altro e come estrarre parti di stringhe

SISTEMA

Il browser sul desktop . . . . **pag. 64**
Progettare un interfaccia è spesso più facile per il Web che per le applicazioni standalone. Allora perchè non unire le due cose progettando software che gira sul tuo computer ma ha una grafica HTML?

Appuntamenti: gestiamoli in Java **pag.70**
Vi è mai capitato di utilizzare la funzione calendario di Outlook? Sapevate che quando invitate qualcuno ad una riunione utilizzate uno standard? In questo articolo vedremo come accedere ai calendari

Facciamo leggere i PDF all'Ipod **pag.74**
In questo articolo cercheremo di utilizzare Java e una famosa libreria per aumentare le potenzialità del nostro Ipod, trasformando i file PDF in documenti leggibili sul nostro lettore multimediale

Programmare .Net con Python **pag.79**
La diffusione di Python come linguaggio di programmazione ha quasi raggiunto quella di .Net, e con Ironpython è ora possibile scrivere script in Python e compilarli per la piattaforma microsoft

Software internazionale **pag.84**
In un'epoca in cui le distanze si sono ormai annullate, è importante produrre applicazioni fruibili in ogni parte del mondo. Il principio base è la localizzazione. vediamo come scrivere codice multilingua

Creare report RTF con Asp.Net 2.0 . . . . **pag.91**
La creazione di reportistica è sempre stata demandata a pacchetti esterni. Ma con RTF si può implementare in pochi e semplici passi. soprattutto si può utilizzare un formato universale almeno quanto PDF

Tenere la memoria sotto controllo **pag.96**
Concludiamo il viaggio nel paese degli Smart Pointers, analizzando quelli più avanzati offerti dalla libreria Boost. La conoscenza di questi elementi è un obbligo per ogni programmatore C++

CORSI BASE

Eclipse nel mondo reale

pag. 103

Che usiate eclipse per sviluppare progetti open source o per il vostro lavoro in azienda, le sue integrazioni con i Tool più usati del mondo Java possono facilitarvi la vita. Vediamo quali sono gli strumenti a disposizione

SOLUZIONI

Ant colony optimization

pag. 110

Osservare il comportamento "sociale" di alcuni animali e insetti può essere un utile insegnamento per riprodurlo con un modello argoritmico. Si tratta di una nuova frontiera dello sviluppo: la Swarm intellingence

RUBRICHE

Gli allegati di ioProgrammo **pag. 6**
Il software in allegato alla rivista

Il libro di ioProgrammo **pag. 8**
Il contenuto del libro in allegato alla rivista

News **pag. 14**
Le più importanti novità del mondo della programmazione

Software **pag. 108**
I contenuti del CD allegato ad ioProgrammo.

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

http://forum.ioprogrammo.it

Versione BASE

RIVISTA + CD-ROM in edicola

TURBO C++ PER WINDOWS 2006

C++ Secondo Borland

Borland è una delle software house storiche nel mercato dello sviluppo. Non si dimentica che proprio Borland ha sviluppato uno dei primi linguaggi completamente ad oggetti: l'object Pascal già a metà degli anni 90. Allo stesso modo è da attribuirsi a Borland anche l'avvento degli IDE Rad con il mitico Delphi. Per qualche anno si era allontanata dal mercato dei tool di sviluppo rivolti alla produttività individuale e solo di recente è ritornata con prepotenza ad occupare questo settore. La nuova linea di prodotti prende il nome di Turbo e vi appartengono Turbo C#, Turbo Delphi, e anche questo meraviglioso Turbo C++ per Windows. Compilatore ultraveloce dotato delle caratteristiche che da sempre hanno fatto grande i prodotti Borland. Prima di tutto una forte attitudine all'uso dei "componenti", ve ne sono circa 200 inclusi in questo pacchetto e poi un'IDE straordinariamente ricco, affidabile e stracolmo di features che facilitano la vita al programmatore. A fare da contorno a tutto questo un compilatore che produce codice altamente ottimizzato e veloce.

Prodotti del mese

PHP 5.2.0

Il linguaggio di scripting del web

Sono tre le colonne portanti di Internet: PHP, APACHE e MySQL. Certo la concorrenza è forte. Asp.NET e SQL Server avanzano con celerità, ma a tutt'oggi non si può affermare che i siti sviluppati in PHP costituiscano la stragrande maggioranza di Internet. Quali sono le ragioni del successo di cotanto linguaggio? Prima di tutto la completezza. PHP ha di base tutto quello che serve ad un buon programmatore, raramente è necessario ricorrere a librerie esterne, e quando è proprio indispensabile farlo esistono comunque una serie di repository che rendono tutto immediatamente disponibile ed in forma gratuita. Il secondo punto di forza sta nella sua capacità di poter essere utilizzato sia in modo procedurale che nella sua forma ad oggetti certamente più completa e funzionale

[pag.108]

Phalanger 2.0

PHP in tecnologia .NET

Ok, siete dei programmatori PHP ma vorreste che il vostro sito girasse in tecnologia .NET. Sapete che il miglio modo per far girare un sito sotto IIS sfruttandone tutte le potenzialità è scrivere codice .NET. Come fate? Facile! La soluzione esiste e si chiama Phalanger. Scrivete il vostro codice in PHP ma in realtà ottenete una pagina compilata in .NET. E' necessario installare qualcosa sul server, ma i risultati sono entusiasmanti!

Certo c'è da installare qualcosa sul server e ancora il supporto non è completo tuttavia si tratta di una tecnologia da sperimentare sia per le prestazioni che è possibile ottenere sia per le opzioni di sicurezza con cui è possibile blindare l'applicazione potendo sfruttare a pieno tutti i meccanismi di protezione di IIS che sicuramente sono raffinati

[pag.109]

Ruby 1.8.5

Il vero nuovo che avanza

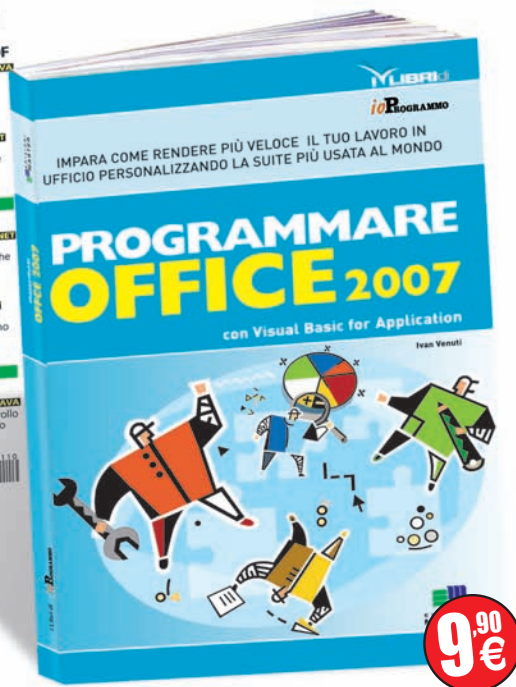
In America è il linguaggio che maggiormente ha scalato i vertici delle classifiche di utilizzo nell'ultimo anno. In Italia sta giungendo rapidamente come un ciclone a fare capolino nel panorama dello sviluppo. Bello, elegante, con una curva di apprendimento praticamente nulla, si tratta di un linguaggio di scripting che ottimamente si presta ad essere utilizzato sia sul Web sia in modo standalone. Recentemente ce ne siamo occupati in un bell'articolo di Paolo Perrotta, che mostrava come utilizzarlo per programmare un plugin per Skype messenger. Il linguaggio è elegante e raffinato, fa un uso intensivo delle liste, ma anche della programmazione ad oggetti. Si presta bene allo sviluppo di qualunque tipo di applicazione e recentemente proprio un framework basato su Ruby: RubyOnRails ha vinto il premio come miglior framework per lo sviluppo web

[pag.109]

Nemerle 0.9.3

L'emergente simile al C#

Di tanto in tanto in programmazione arrivano delle novità assolute che inizialmente sfuggono al grande pubblico per poi diffondersi lentamente nel corso del tempo ed arrivare alla piena maturità in un periodo di tempo abbastanza lungo. E' stato il caso di Ruby molti anni fa e di Python ancora prima. E'il caso di Boo e di Nemerle adesso. Nemerle è un linguaggio nato per la piattaforma .NET con una sintassi molto simile al C#. Si tratta di un linguaggio fortemente tipizzato ed orientato agli oggetti ma con la possibilità quando è il caso di essere utilizzato nella sua forma procedurale e da questo punto di vista segue i concetti cari al PHP. Altro aspetto interessante è quello di poter utilizzare le macro. Si tratta di un progetto interessante che merita una particolare attenzione. Nonostante la sua giovane età dispone di caratteristiche uniche

[pag.109]

I contenuti del libro

Programmare Office 2007

Finalmente ci siamo Office 2007 è alle porte con il suo ricco carico di novità. Si tratta della suite più usata in ufficio e che in molti casi sostiene l'intera procedura di automazione aziendale. Certo, in molti la utilizzano al minimo delle potenzialità, tuttavia Office 2007 è molto di più di un semplice editor, un database e un foglio elettronico. Si tratta di un vero insieme di programmi che comunicano e interagiscono fra loro. Dall'archiviazione dei dati alla produzione dei documenti tutto può essere automatizzato con questa Suite, e quando le normali funzioni non bastano ecco intervenire Visual Basic for Application. Un linguaggio completo che consente di estendere e personalizzare tutti gli elementi che compongono questa suite secondo le esigenze che caratterizzano le varie aziende o il singolo individuo. Conoscere le tecniche di programmazione alla base di Office significa aumentrae di molto la propria produttività individuale e il processo di produzione dell'azienda

IMPARA COME RENDERE PIÙ VELOCE IL TUO LAVORO IN UFFICIO PERSONALIZZANDO LA SUITE PIÙ USATA AL MONDO

- **Introduzione al Visual Basic for Application**
- **VBA e finestre grafiche**
- **L'architettura di Office**
- **Ribbon e altre novità**

GLI ALLEGATI DI IOPROGRAMMO

Questo mese vi presentiamo tre Webcast sul nascituro Windows Vista e sul nuovo .NET Framework 3.0

Windows Vista e .NET Framework 3.0: overview

Questo Webcast illustra le novità di Windows Vista per gli sviluppatori, con l'obiettivo di fornire una panoramica del nuovo sistema operativo e Framework di Microsoft.

Speaker: Fabio Santini

Architettura del sistema operativo

Windows Vista è l'ultima versione della famiglia dei sistemi operativi derivata da Windows NT e, dal punto di vista architetturale, presenta il maggior numero di cambiamenti rispetto alle precedenti. Spesso le novità non influiscono sul comportamento delle applicazioni finali , ma sono importanti per aumentare il livello di sicurezza e di prestazioni del sistema. Windows Vista offre inoltre nuove e interessanti funzionalità alle applicazioni scritte per supportare questa versione del sistema operativo.

Speaker: Marco Russo

Il nuovo supporto per gli RSS

RSS (Really Simple Syndication) è un formato XML molto comodo per descrivere news, annunci e contenuti dinamici. Sempre più aziende e siti Web stanno adottando questo formato per pubblicare e consentire il monitoraggio dei loro siti Internet. Windows Vista, Internet Explorer 7 e Outlook 2007 supportano RSS nativamente. Microsoft permette inoltre di sviluppare applicazioni integrate con RSS mediante Windows RSS Platform. In questo Webcast vedremo sia i principi base di RSS sia le funzionalità di Windows RSS Platform.

Speaker: Paolo Pialorsi

msdn WEBCAST

PERCORSI FORMATIVI

Per chi desidera approfondire questi temi, sono disponibili sul sito MSDN una serie di Webcast prodotti dagli esperti Microsoft e dedicati agli sviluppatori, fra cui:
• Windows Presentation Foundation • WPF Windows Communication Foundation • WCF Security & privacy • Internet Explorer 7 • Developing Gadgets for Windows Toolbar • Microsoft Expression Interactive Designer • Come testare la compatibilità delle proprie applicazioni

Visita
www.microsoft.com/italy/msdn/risorsemsdn/windowsvista/default.mspx

FAQ

Cosa sono i Webcast MSDN?

MSDN propone agli sviluppatori una serie di eventi gratuiti online e interattivi che approfondiscono le principali tematiche relative allo sviluppo di applicazioni su tecnologia Microsoft. Questa serie di "corsi" sono noti con il nome di Webcast MSDN

Come è composto tipicamente un Webcast?

Normalmente vengono illustrate una serie di Slide commentate da un relatore. A supporto di queste presentazioni vengono inserite delle Demo in presa diretta che mostrano dal vivo come usare gli strumenti oggetto del Webcast

Come mai trovo riferimenti a chat o a strumenti che non ho disponibili nei WebCast allegati alla rivista?

La natura dei WebCast è quella di essere seguiti OnLine in tempo reale. Durante queste presentazioni in diretta vengono utilizzati strumenti molto simili a quelli della formazione a distanza. In questa ottica è possibile porre domande in presa diretta al relatore oppure partecipare a sondaggi etc. I WebCast riprodotti nel CD di ioProgrammo, pur non perdendo nessun contenuto informativo, per la natura asincrona del supporto non possono godere dell'interazione diretta con il relatore.

Come mai trovo i WebCast su ioProgrammo

Come sempre ioProgrammo cerca di fornire un servizio ai programmatori italiani. Abbiamo pensato che poter usufruire dei Webcast MSDN direttamente da CD rappresentasse un ottimo modo di formarsi comodamente a casa e nei tempi desiderati. Lo scopo tanto di ioProgrammo, quanto di Microsoft è infatti quello di supportare la comunità dei programmatori italiani con tutti gli strumenti possibili.

Su ioProgrammo troverò tutti WebCast di Microsoft?

Ne troverai sicuramente una buona parte, tuttavia per loro natura i webcast di Microsoft vengono diffusi anche OnLine e possono essere seguiti previa iscrizione. L'indirizzo per saperne di più è: http://www.microsoft.it/msdn/webcast/msdn segnalo nei tuoi bookmark. Non puoi mancare.

L'iniziativa sarà ripetuta sui prossimi numeri?

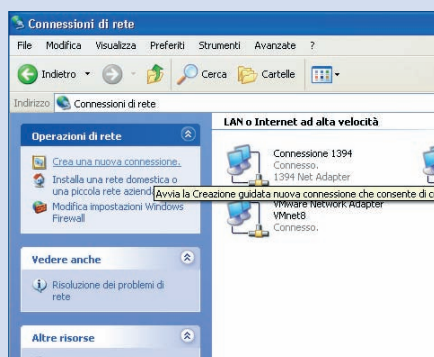
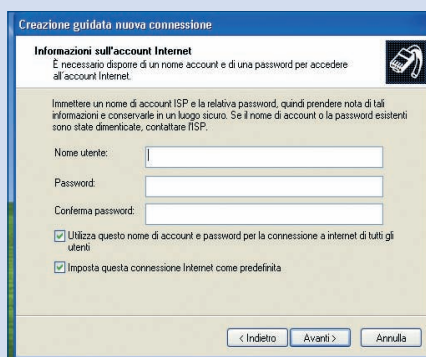
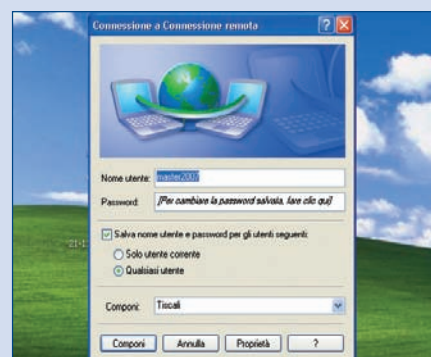
Sicuramente si.

Online con Tiscali Easy

Numero unico per tutta l'Italia e nessuna registrazione. Il modo più semplice e veloce per entrare in Internet risparmiando

Sei spesso lontano da casa e hai necessità di collegarti a Internet ovunque ti trovi? Desideri salvaguardare la tua privacy navigando in modo anonimo? Non hai voglia di perdere tempo in lunghe registrazioni per creare un nuovo account? Niente paura, Tiscali ha pensato anche a te! Con Tiscali Easy arriva un sistema tutto nuovo di collegarsi a Internet. Non sarà più necessario creare un nuovo account e fornire i propri dati personali, potrai navigare collegandoti con un unico numero di telefono (7023456789) da tutta Italia e soprattutto utilizzando i dati di accesso forniti direttamente da Tiscali (UserID e Password presenti sulla scheda), quindi non collegabili in alcun modo a te. Insomma, con Tiscali Easy, otterrai in un colpo solo riservatezza dei dati, risparmio del tempo necessario alla creazione di un abbonamento e tariffe vantaggiose. I costi di connessione sono simili a quelli di un classico abbonamento gratuito: 1,90 cent. per i primi 10 minuti e 1,72 cent. per quelli successivi. Per risparmiare ulteriormente è possibile connettersi a Internet durante le ore serali o nei giorni festivi al costo di 1,09 cent. per i primi dieci minuti e 0,98 cent. per quelli successivi. L'unico requisito richiesto per poter eseguire la connessione è un modem correttamente installato. Poiché si tratta di una normale connessione analogica è possibile utilizzare i tool di accesso remoto integrati in Windows

IN RETE CON TISCALI

Nessuna registrazione basta creare una nuova connessione e inserire i dati di accesso forniti da Tiscali

1 NUOVA CONNESSIONE Portiamoci nel pannello di controllo, e selezioniamo l'icona connessioni di rete. In alto a sinistra selezioniamo "nuova connessione" e prepariamoci a seguire il wizard che comparirà

2 DATI DI ACCESSO Seguiamo il Wizard fino a quando non ci vengono chiesti i dati per l'autenticazione. E' sufficiente inserire quelli presenti nella card allegata a questo numero di ioProgrammo: master2007/tiscali

3 ACCESSO A INTERNET Connettersi è semplicissimo, troveremo una nuova icona all'interno del pannello di controllo alla voce connessioni. Bastano due click ed il gioco è fatto sarete connessi ovunque vi troviate

News

SQL SERVER VS ORACLE: È GUERRA!

Il fuoco alle polveri lo ha dato un recente studio condotto da NGSS – Next Generation Security Software, reso recentemente noto all'indirizzo http://www.databasesecurity.com/dbsec/comparison.pdf. Lo studio ha preso in considerazioni le patch rilasciate per i due noti database nel corso degli ultimi 6 anni. La palma del vincitore è andato a MS SQL Server che avrebbe rilasciato 59 patch contro le 233 del database di Oracle che addirittura avrebbe ancora 49 fal-

le aperte nel proprio sistema. Gli analisti non si sono dimostrati molto propensi a dare fiducia a questo rapporto contestando che SQL Server ha solo recentemente conquistato fette di mercato rilevanti e che non si sono considerate a sufficienza le falle relative al SQL Injection sul database di Microsoft. I portavoce di Oracle si sono limitati ad affermare che la ricerca è stata troppo semplicistica e che il livello di sicurezza andrebbe misurato anche in base agli effettivi scenari d'uso e agli ambienti in cui i sistemi sono installati

GRATIS LA GUI DI OFFICE 2007

Ribbon, questo è il nome con cui Microsoft identifica l'interfaccia grafica di Office 2007. Gli sviluppatori che volessero creare applicazioni con il look & feel di Ribbon potranno farlo a patto di rispettare la voluminosa licenza gratuita di appena 120 pagine con cui Microsoft certificherà le cose che si possono e non si possono fare. In questo modo da un lato MS protegge proprio lavoro, dall'altro stimola gli utenti a creare interfacce omogenee sia fra di loro sia con il sistema operativo evitando la nascita di GUI ibride.

PACE FATTA FRA MICROSOFT E NOVELL

Per lungo tempo sono stati rivali, impegnati in più di una causa legale relativa alla reciproca infrazione di più di un brevetto. Improvvisamente tutto è cambiato. Microsoft e Novell hanno raggiunto un accordo per la cooperazione e l'interoperabilità. I termimi dell'accordo sembrano a prima vista abbastanza semplici. In realtà ad uno sguardo più approfondito emerge qualche insidia che ha preoccupato non poco la comunità OpenSource. D'altra parte un segmento degli utilizzatori di Linux si è dichiarata entusiasta. Questo equilibrio di posizioni mostra quanto in realtà ci si muova su una linea sottile che rende lo storico accordo sufficientemente complesso. In sostanza Microsoft consiglierà SuSE Linux a tutte quelle aziende che fanno uso di soluzioni miste. Allo stesso modo MS garantirà il supporto a tutti e due i sistemi operativi. La parte del leone la farà la virtualizzazione, Microsoft certificherà che i sistemi SuSE possano girare in modo virtualizzato in ambiente Windows, lo stesso farà SuSE nei riguardi dei sistemi operativi prodotti dalla casa di Redmond. Una seconda parte dell'accordo prevede la reciproca protezione legale. Ovvero le due parti si impegnano a non intraprendere azioni legali l'una nei confronti dell'altra in relazione a presunte infrazioni su alcuni brevetti. Una terza parte prevede la creazione di laboratori congiunti attraverso i quali si possano sviluppare tecnologie che innalzino la qualità della collaborazione fra i due ambienti. Certamente questo accordo portata una ventata d'aria nuova nel mercato dei server, dove fino ad ora era stata RedHat a detenere le maggiori quote di mercato

RUBY E PYTHON SEMPRE PIÙ IN ALTO

TIOBE (*http://www.tiobe.com/*) ha appena pubblicato la consueta classifica dei linguaggi più gettonati del momento. Questa speciale "hit parade" viene stilata sulla base di un elevato numero di parametri, fra cui la disponibilità di esperti di un certo linguaggio, il numero di progetti disponibili, l'adozione in ambiti di grande rilievo. Si tratta ovviamente di una classifica che non rispetta dei canoni di oggettività incontestabili, tuttavia fornisce la misura di quanto un linguaggio stia incontrando i favori del mercato oppure no. E' un buon indice per chi necessita di un consiglio per iniziare o per le aziende che devono decidere in quale segmento dello sviluppo investire. Per Novembre 2006 vede in testa Java che tuttavia fa registrare un 1,87% in meno rispetto allo stesso periodo dell'anno scorso. A seguire C e C++ con il primo in netto calo e il secondo in crescita di appena un 1.25%. Sale in quarta posizione Visual Basic che guadagna addirittura un 1.89% rispetto all'anno precedente. Anche Python in netta crescita, con un +0.87% si posiziona in settima posizione rubando il posto addirittura al blasonatissimo C#. A sorprendere più di tutti sono Ruby e D, rispettivamente in dodicesima e quattordicesima posizione con un incremento di otto posizioni per il primo e addirittura di tredici per il secondo! E' vero che questa classifica va letta e adattata anche secondo la propria percezione del mercato, tuttavia se può fornire un'indicazione appare evidente quanto i linguaggi di scripting multipiattaforma stiano velocemente conquistando posizioni significative. Ad incidere su questa tendenza è probabilmente la semplicità con cui questi linguaggi si integrano nei diversi sistemi

ESPLORANDO IL .NET FRAMEWORK 3.0

IL NUOVO MODELLO DI PROGRAMMAZIONE MANAGED PER WINDOWS. COMBINA LE CARATTERISTICHE DI .NET 2.0 CON LE NUOVE TECNOLOGIE PER LO SVILUPPO DI INTERFACCE GRAFICHE, COMUNICAZIONE DI RETE, E IL SUPPORTO DEI PROCESSI AZIENDALI.

Lo scopo di una qualsiasi software house è quello di rilasciare il miglior prodotto software possibile, e poi di mantenerlo apportando miglioramenti e nuove funzionalità, oltre a correggere gli errori ed i bug presenti in una versione precedente dello stesso prodotto. Con .NET Framework 1.1, 2.0, Microsoft ha seguito senza dubbio la strada dell'innovazione e del miglioramento rilasciando una piattaforma sempre più stabile e che aumenta proporzionalmente la produttività dello sviluppatore Windows. .NET Framework 3.0 prosegue lungo la stessa strada apportando una notevole mole di novità, di nuove funzionalità, ed un ambiente di sviluppo ancora più produttivo. La nuova release è stata annunciata nella sua veste definitiva da Microsoft lo scorso 11 Novembre.
.NET 3.0 può essere pensato come ad un .NET 2.0 con l'aggiunta delle estensioni dell'ex WinFX, e cioè Windows Communication Foundation, Windows Presentation Foundation, Windows Workflow Foundation Windows Cardspace.
Mentre C# 3.0, ASP.NET 3.0 e compagnia faranno parte invece di .NET 3.5, forse!

REQUISITI

Conoscenze richieste
media conoscenza del .NET Framework 2.0

Software
Windows XP SP2, Windows Server 2003, Windows Vista; Visual Studio 2005 Standard o superiore; .NET Framework 3.0 Runtime Components (già inclusi in Vista); Visual Studio Extensions for Windows Workflow Foundation, Visual Studio Extensions for Windows Presentation Foundation e Windows Communication Foundation.

Impegno

Tempo di realizzazione

ARCHITETTURA DI .NET 3.0

.NET 3.0 è costruito al di sopra di .NET 2.0. La figura 1 mostra una vista di insieme della nuova architettura.

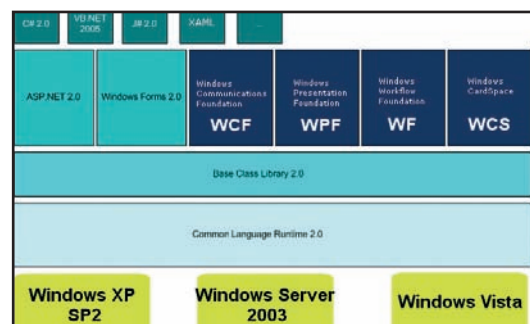



Fig. 1: Lo schema a blocchi dell'architettura

I componenti che costituiscono .NET Framework 3.0, si pongono al di sopra e funzionano in collaborazione con il .NET framework 2.0. Ciò significa fra l'altro che le "vecchie" applicazioni scritte per .NET 2.0 continueranno a funzionare senza alcuna modifica anche dopo aver installato il nuovo framework, ciò perché il CLR non ha subito alcuna modifica, così come sono rimaste invariate le librerie della Base Class Library. Nessuna paura di incompatibilità quindi.
Sarà poi ancora possibile scrivere applicazioni che non passano attraverso le nuove funzioni di .NET 3.0, ma che lavorano soltanto con .NET 2.0, oppure, soprattutto per i nostalgici, applicazioni native, che utilizzano direttamente l'interfaccia Win32.
Molte parti del .NET Framework 2.0 saranno superate e sostituite dalle corrispondenti parti dedicate di .NET 3.0, come vedremo nei prossimi paragrafi, ma le tecnologie di .NET 2.0 rimangono una parte fondamentale della nuova release.

WINDOWS WORKFLOW FOUNDATION (WF)

Un workflow è un concetto molto semplice: una sequenza di attività eseguite in un dato ordine.
Windows Workflow Foundation (WF) fornisce una tecnologia comune alle applicazioni che hanno la necessità di implementare tale logica. Molte aziende ormai hanno un proprio processo di sviluppo, facilmente pensabile come un workflow, e Microsoft ha pensato proprio a questo nel momento in cui ha deciso di introdurre WF in .NET 3.0.
Tra l'altro anche altri software di Microsoft utilizzano o utilizzeranno WF: Office System 2007 fa parte di questi. Dato che esistono e possono esistere decine, centinaia o migliaia di diversi workflow con altrettanti diversi requisiti, realizzare una tecnologia come WF, che possa gestirli tutti, ha richiesto un certo sforzo di astrazione. La soluzione è stata quella di considerare il concetto più generale possibile di workflow. In tal modo un workflow di WF è un insie-

me di attività eseguite da un motore. Ogni attività è rappresentata da una classe che contiene il codice necessario ad eseguirla. Le attività possono così essere riutilizzate in workflow diversi. La figura 2 mostra in maniera grafica tale concetto:

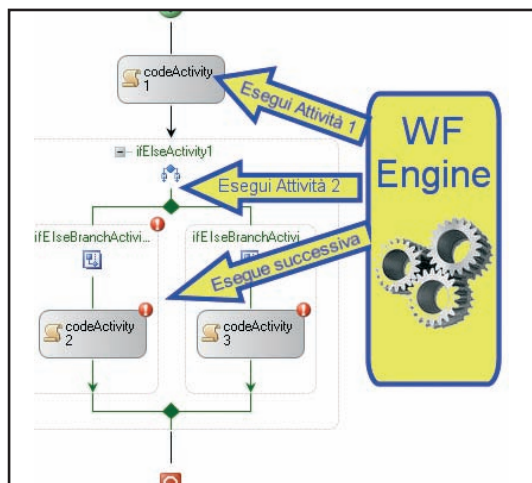



Fig. 2: ***esecuzione di un workflow***

Visual Studio con le estensioni per WF consente di creare un workflow in maniera grafica, e generare un corrispondente codice in un nuovo linguaggio di markup, XOML (Extensible Object Markup Language), oppure, se si preferisce direttamente in codice sorgente C#.

WINDOWS COMMUNICATION FOUNDATION (WCF)

Qualunque sia il tipo di applicazione da sviluppare, capita abbastanza di frequente, soprattutto in ambito enterprise, la necessità di stabilire una comunicazione fra diverse applicazioni.
Dopo anni in cui ogni attore del mondo IT cercava di imporre la propria tecnologia o il proprio software, tutti i maggiori protagonisti si sono messi d'accordo per cercare un protocollo comune. Ciò ha portato ad esempio alla nascita di SOAP, dei Web Service, ed alla possibilità di intercomunicazione fra piattaforme completamente differenti, basti pensare a J2EE e .NET che oggi parlano fra loro in maniera nettamente più semplificata rispetto a qualche anno fa.
Microsoft, con Windows communication Foundation (WCF), in origine chiamato in codice Indigo, introduce una interfaccia di programmazione comune per ogni tipo di comunicazione. In tal senso WCF potrà realizzare, da solo, quello che oggi fanno altre tecnologie, come i servizi web, asp.net, .NET Remoting, gli Enterprise Services e così via.
In .NET 3.0 applicazioni che prima utilizzavano una o più delle precedenti tecnologie, potranno utilizzare solo WCF, semplificando di molto il tutto.

Windows Communication Foundation (WCF) è un runtime ed un insieme di API per creare sistemi che spediscono messaggi fra servizi e client, il messaggio è il concetto fondamentale. WCF può così essere utilizzato per creare applicazioni che comunicano con altre applicazioni sullo stesso sistema, oppure su sistemi diversi che risiedono altrove, tramite internet. La figura 3 mostra l'architettura di alto livello di WCF.

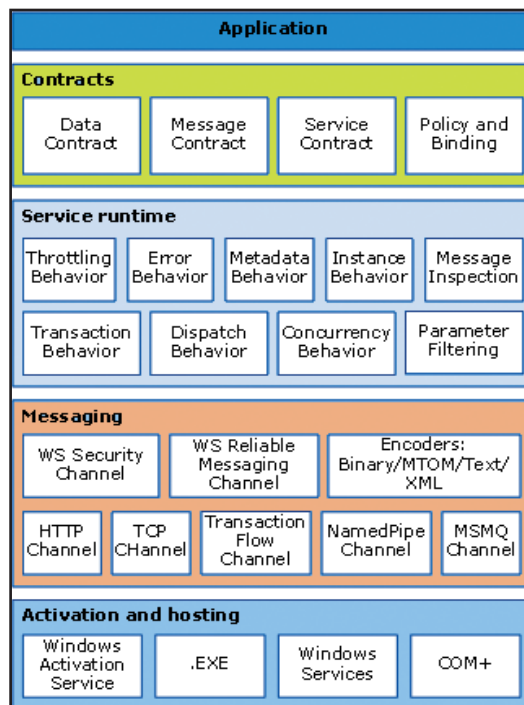



Fig. 3: ***L'architettura di WCF***

INSTALLAZIONE DI .NET 3.0

È possibile installare .NET Framework 3.0 nei sistemi operativi seguenti:
- Microsoft Windows 2003 Server Service Pack 1 (SP1)
- Windows XP SP2
Windows Vista viene fornito già con .NET 3.0, e dunque può già eseguire applicazioni .NET 3.0 Enabled. Per gli altri sistemi operativi sono stati rilasciati i .NET 3.0 Runtime Components (cioè il Redistributable Package), disponibile anche in italiano, basta cercarli su MSDN nella sezione download.
Microsoft ha poi rilasciato il nuovo SDK di Windows, dedicato naturalmente a Windows Vista, e contenente naturalmente il .NET Framework 3.0 SDK. Si tenga presente che il solo SDK non è sufficiente ad eseguire applicazioni scritte in .NET 3.0, quindi i Runtime Components sono in ogni caso richiesti.
Oltre a tali download sono poi disponibili due estensioni per Visual Studio 2005, vale a dire quella per il supporto di Windows Workflow Foundation e quella, ancora in CTP al momento dell'articolo, per sviluppare con WPF e WCF.
Chi avesse già installato in precedenza una versione Beta, dovrà prima rimuoverla utilizzando l'apposito Uninstall Tool: http://www.microsoft.com/downloads/details.aspx?FamilyId=AAE7FC63-D405-4E13-909F-E85AA9E66146&displaylang=en
Per eseguire applicazioni scritte per .NET 3.0 è necessario installare il pacchetto .NET Framework 3.0 Runtime Components,

I contratti definiscono i vari aspetti del sistema di messaggistica, come si può comunicare con esso, cioè quali operazioni si possono chiamare e quali parametri possono essere passati.
Lo strato Service Runtime contiene i diversi behaviour, cioè definisce il comportamento del servizio nelle varie situazioni possibili durante il suo ciclo di vita, quanti messaggi possono essere processati, cosa succede in caso di errore, come e quando i messaggi vengono processati e così via.
Lo strato di messaging è composto da canali, cioè i componenti che processano i messaggi. Esistono due tipologie di canali, quelli di trasporto e quelli di protocollo: i primi si occupano della lettura e della scrittura dei messaggi sulla rete, i secondi dell'interpretazione dei protocolli di comunicazione.
Ogni servizio è alla fin fine un programma, che può essere eseguito come una qualsiasi applicazione, oppure ospitato ed attivato da un'applicazione separata. Lo strato Activation and hosting si occupa di questo.
WCF supporta diversi tipi di comunicazione a messaggi, per default tutte del tipo Request-Reply, ma con la possibilità naturalmente di utilizzarle in diverse maniere. Ad esempio, capita spesso che un client debba solo invocare un'operazione remota e non gli importa del risultato, quindi basta una semplice comunicazione One Way, altre volte è necessaria invece una infrastruttura in cui il servizio possa invocare una funzione del chiamante, e si ha la cosiddetta operazione Callback, o ancora possono essere necessarie comunicazioni con sottoscrizioni di eventi o di tipo publish-subscribe. WCF permette tutto ciò ed altro ancora.
Il concetto fondamentale di client che accede ed utilizza un servizio è mostrato in figura 4.

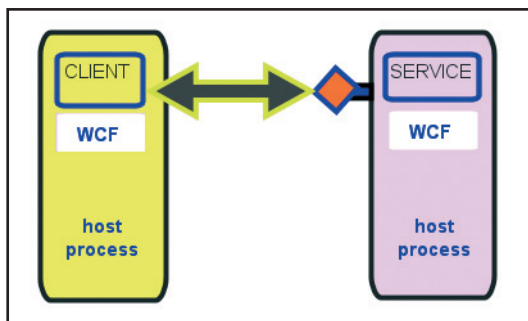



Fig. 4: *Client che utilizza un servizio con WCF*

WINDOWS CARDSPACE (WCS)

Quando gli utenti di un'applicazione, sia essa Web, o Windows, accedono ad essa, si ha una sorta di trasferimento della loro identità digitale, che può essere costituita dal loro nome utente, dalla password, o di altre informazioni ancora più sensibili, con conseguenti ed intuibili problemi di sicurezza.
Windows Cardspace, prima conosciuto come InfoCard, si occupa del trattamento delle diverse identità digitali degli utenti.
Esso è un sistema per creare delle relazioni fra un sito web o un servizio ondine e l'utente. WCS fornisce un modo per permettere a tali servizi di richiedere le informazioni sull'utente, all'utente di essere sicuro dell'identità del sito, di gestire le informazioni per mezzo di card, e di inviarle a tali servizi, senza la necessità di tenere a mente decine di nomi utenti, di password, di codici di accesso e così via. Basta creare una nuova Card, dare un nome facilmente associabile ad un dato servizio, e si potrà utilizzare questa per accedere al servizio stesso ogni volta che ce n'è bisogno.
Dopo l'installazione di .NET 3.0, il pannello di controllo di Windows conterrà una nuova icona, Windows Cardspace appunto, dal quale gestire le proprie Card. La figura 5 mostra la schermata di creazione di una nuova identità.

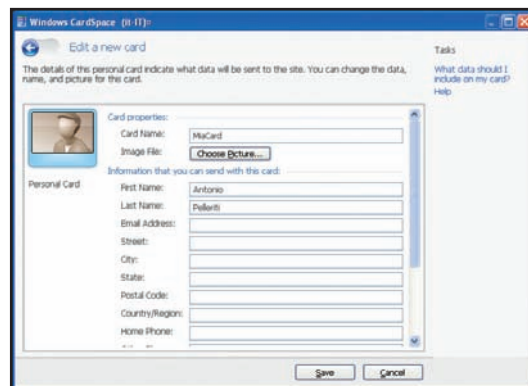



Fig. 5: *Creazione di una Card per Windows Cardspace*

WINDOWS PRESENTATION FOUNDATION (WPF)

Qualunque sia la complessità di un'applicazione, sia essa un'applicazione basata su Workflow, su Servizi, quello che l'utente vede e che spesso ritiene ancora più importante è l'interfaccia utente. Windows Presentation Foundation (WPF), in origine Avalon, è la tecnologia di .NET 3.0 destinata alla creazione di interface grafiche evolute e coerenti anche in diversi ambienti, client e tecnologie. Molte applicazioni sono ormai accessibili sia come classiche applicazioni Windows, ma anche fruibili da un browser, ed è quindi spesso necessario un doppio sforzo di sviluppo e manutenzione dell'interfaccia grafica. WPF promette di rendere più facile la vita agli sviluppatori in tal senso, ed allo stesso tempo, offre una

immensa varietà di componenti grafici, con supporto per video, animazioni, grafica 2D e 3D e così via. WPF per certi versi soppianta, ma non sostituisce, Windows Forms 2.0, ancora presente. L'innovazione principale è forse quella di poter usare grafica vettoriale, e creare contenuti più facilmente fruibili dagli utenti, su diverse piattaforme. Basti pensare al classico problema del ridimensionamento di una finestra su monitor con diverse risoluzioni: con grafica vettoriale il problema non si pone.
Tutto ciò che WPF promette è realizzabile grazie ad un nuovo linguaggio di markup, chiamato XAML (Extensible Application Markup Language), derivato da XML, con cui vengono costruiti e definiti i building block delle interfacce grafiche. Per esempio per creare una finestra con un singolo pulsante, basterà creare un file con il seguente contenuto XAML:

```
<Window
xmlns="http://schemas.microsoft.com/winfx/2006/xa
                                  ml/presentation">
        <Button>Hello WPF!</Button>
</Window>
```

WPF interpreta e traduce gli elementi presenti nel codice XAML, in corrispondenti classi .NET.
Naturalmente Visual Studio, tramite le estensioni dedicate a WPF, consente di creare XAML non solo scrivendo direttamente il codice XML, ma passando attraverso l'aggiornato designer dedicato ad esso.
Creare una interfaccia utente utilizzando XAML ha diversi vantaggi rispetto alle tecnologie tradizionali: XAML è spesso molto più facile da leggere, ed espressivo quindi rispetto a del codice C#. XAML consente di separare la logica dell'applicazione dalla definizione dell'interfaccia grafica, e quindi consente un maggior riutilizzo del codice scritto oltre che una maggiore robustezza dell'insieme. Inoltre in tale maniera un grafico potrà lavorare all'interfaccia grafica, mentre uno sviluppatore si occupa del lato di logica.

SVILUPPARE CON .NET 3.0

Per installare e dotare la vostra macchina di ciò di cui si ha bisogno per sviluppare applicazioni per e con .NET Framework 3.0, date un'occhiata al box dedicato, in questo stesso articolo. Dopo aver installato in particolare le estensioni per Visual Studio 2005, fra i template disponibili per la creazione di un nuovo progetto saranno presenti sia quelli per la creazione di applicazioni WPF ed un template per la creazione di una libreria WCF al di sotto della categoria Visual C#, e sei nuovi template per la creazione di applicazioni WF raggruppati nella categoria Workflow (figura 6).

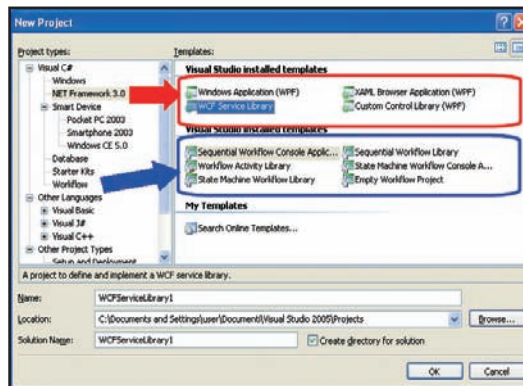



Fig. 6: *i nuovi template per WPF, WCF, WF*

HELLO WORKFLOW

Il primo passo nella creazione di un'applicazione WF, supponendo di aver già definito un workflow o di averne uno pronto da implementare, è la creazione di un progetto Visual Studio. Dalla lista dei template disponibili (figura 7) dopo l'installazione delle estensioni per Windows Workflow, si selezioni il tipo Sequential Workflow Console Application, si assegni un nome al progetto, e poi si faccia clic su OK.

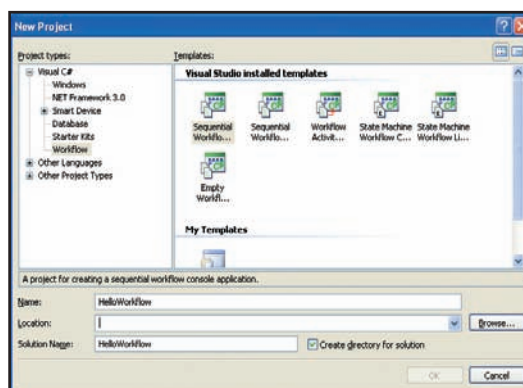



Fig. 7: *creazione del progetto WF*

Il progetto conterrà un oggetto Workflow1.cs, che può essere per il momento eliminato, perché ne verrà creato uno da zero. Facendo clic con il tasto destro quindi sul progetto nel Solution Explorer, si selezioni la voce Add->Sequential Workflow, e dalla schermata seguente (figura 8) il tipo Sequential

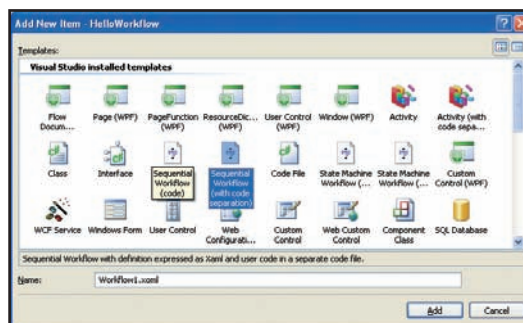



Fig. 8: *aggiunta di un WF*

Workflow (with code separation), dopo aver eventualmente assegnato un nome al workflow, e fatto clic su Add, verrò creato un file Workflow1.xoml, che è il file contenente il codice di markup in linguaggio xoml, ed un file Workflow1.xoml.cs, contenente il code beside dello stesso Workflow.
Nel designer di Visual Studio il Workflow sarà visualizzato in maniera grafica, e dalla toolbox si potrà trascinare un elemento Code per creare una prima attività per il nostro processo (figura 9).

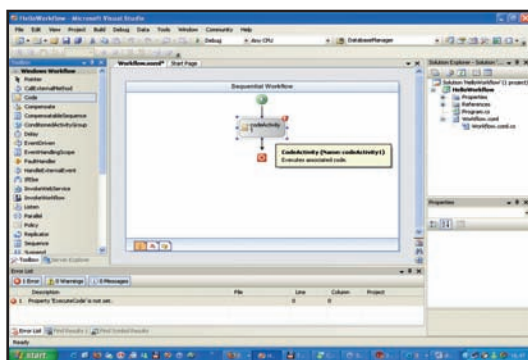



Fig. 9: ***Creare un'attività del Workflow***

Come si noterà dalla figura, la prima attività è contrassegnata da un punto esclamativo, facendo clic su di esso si verrà avvisati che per l'attività stessa non è impostata la proprietà ExecuteCode, cioè non è stato definito il codice che descrive l'attività da eseguire. In questo caso basta fare doppio clic sull'attività, o impostare la proprietà dalla finestra Properties del Solution Explorer.

```
private void codeActivity1_ExecuteCode(object
                                    sender, EventArgs e)
{
        CodeActivity c = (CodeActivity)sender;
        Console.WriteLine("Hello, da '{0}'.\nSono
                      un'istanza della classe {1}.",
          c.Name, c.GetType().ToString());
}
```

Il codice precedente farà stampare sulla console una stringa "Hello", seguita dal nome dell'attività e dal tipo, quando il motore di WF eseguirà l'attività stessa. Per effettuare il debug del primo Workflow basta come al solito premere il tasto F5. Prima si imposti però un breakpoint sull'attività, premendo il tasto F9 (o dal menù contestuale Breakpoint->Insert Breakpoint) dopo averla selezionata nel designer, dovrebbe apparire un pallino rosso sul bordo sinistro. Avviato il debug verrà create un'istanza del Workflow, ed il runtime di Workflow Foundation provvederà ad avviarla. Andando a vedere il codice presente nel metodo Main del file Program.cs, si noterà infatti il seguente:

```
WorkflowInstance instance =
  workflowRuntime.CreateWorkflow(typeof(HelloWorkfl
                                      ow.Workflow));
instance.Start();
```

In debug l'esecuzione si fermerà appena il Workflow dovrà eseguire la codeActivity1, e questa apparirà evidenziata in giallo, allo stesso modo in cui Visual Studio evidenzia il sorgente ad un breakpoint sul codice. Continuando l'esecuzione, sulla console verrà stampata la stringa suddetta, dopodichè il Workflow terminerà.

CONTROLLO DEL FLUSSO

Adesso si proverà a complicare un po' il processo, in maniera da vedere come eseguire una attività od una seconda in base ad una determinata condizione, vale a dire come modellare un classico if/else.
Per aggiungere un blocco if/else basta trascinare sul designer del Workflow, un'attività IfElse su uno dei rami in cui è visualizzato un simbolo +. Il blocco IfElse ha per default due soli rami, ma si possono anche creare più rami semplicemente facendo clic sulla voce Add Branch del menù contestuale del blocco. Per abilitare l'esecuzione di un ramo dovrà essere soddisfatta una data condizione, che puà essere specificata in due diversi modi: via codice o tramite un'espressione che restituisce un valore booleano. In questo esempio si utilizzerà un'espressione. Si supponga di voler eseguire il ramo a sinistra nel caso in cui l'utente sia un nuovo cliente, per esempio per consentirne la registrazione su un sito, mentre quello a destra sarà solo per gli utenti registrati, e quindi per effettuare il login (vedi figura 10).
Aggiungiamo intanto alla classe un campo booleano e la corrispondente proprietà di accesso, che servirà ad indicare se un utente è registrato o meno:

```
private bool m_newUser;
public bool IsNewUser
{
        get { return m_newUser; }
        set { m_newUser = value; }
}
```

CHI HA INCASTRATO WINFX

La numerazione 3.0 può ingannare chi non ha seguito le ultime vicende pre-rilascio della piattaforma, che in origine era stata battezzata con il più esotico nome WinFX, e poi, nonostante le proteste provenienti da diverse parti, cambiata in .NET 3.0. Il nome .NET 3.0 lascerebbe infatti pensare ad una nuova versione del CLR, quindi ad un nuovo compilatore, e magari all'inclusione in tale rilascio delle caratteristiche di C# 3.0, di ASP.NET 3.0 o di un nuovo VB.NET. Niente di tutto questo, .NET 3.0 è in realtà una sorta di .NET 2.5.

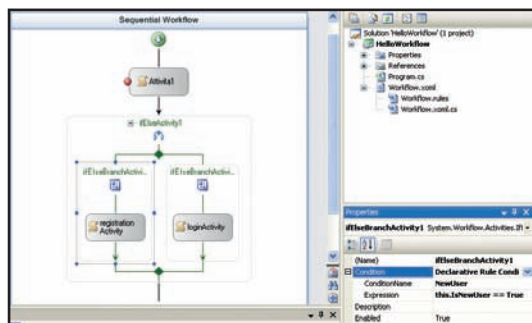



Fig. 10: ***Controllo del flusso delle attività***

Per impostare la condizione su un ramo, si faccia clic su di esso, poi nella finestra delle proprietà su Condition, scegliendo in questo caso la tipologia Declarative Rule Condition. Adesso bisogna assegnare prima un nome alla condizione, per esempio NewUser, e poi costruire l'espressione booleana utilizzando il Rules Condition Editor (figura 11). Per avviarlo basta fare clic sui tre puntini nel campo Expression.

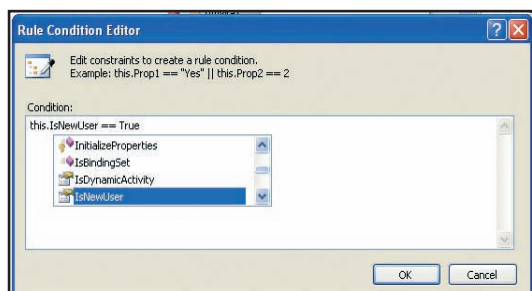



Fig. 11: ***Definizione di una regola per WF***

Nell'esempio basterà verificare che la proprietà IsNewUser è true, per eseguire il ramo di registrazione. Nel caso contrario verrà naturalmente eseguito l'altro ramo.
Nei due rami basterà poi creare le due attività esclusive da eseguire, ed impostare le relative proprietà ExecuteCode, in questo caso si stamperanno solo delle stringhe per verificare che il flusso è quello corretto:

```
private void registrationActivity_ExecuteCode(object
                                      sender, EventArgs e)
{
Console.WriteLine("Procedura di registrazione");
}
private void loginActivity_ExecuteCode(object sender,
                                               EventArgs e)
{
        Console.WriteLine("Procedura di login");
}
```

Non resta che aggiungere del codice per far scegliere all'utente se vuole registrarsi o effettuare il login, nella prima attività, impostare la variabile m_newUser, ed il gioco, oops, il workflow è fatto.

```
private void codeActivity1_ExecuteCode(object
                                      sender, EventArgs e)
{
    Console.WriteLine("Sei un utente registrato?
                                                  (s/n)");
    if (Console.ReadLine() == "s")
        _newUser = true;
    else m_newUser = false;
}
```

CREARE SERVIZI E CLIENT WCF

Dopo averne parlato in maniera teorica e architetturale si affronterà ora l'implementazione di un'applicazione WCF, a partire dalla creazione del servizio completo, inclusa la sua definizione, la codifica, e l'hosting. Il primo passo è in genere la definizione dei contratti e poi implementarli.
Si supponga di voler realizzare un servizio per una pizzeria, a cui un cliente può richiedere l'elenco delle pizze disponibili con i relativi prezzi. Il primo passo sarà quello di rappresentare una pizza, con una classe.
Da Visual Studio 2005, si crei innanzitutto un nuovo progetto, scegliendo come tipologia WCF Service Library, ed assegnando un nome adeguato: per esempio PizzaWCFServiceLibrary. Visual Studio 2005 creerà il progetto ed un file Class1.cs, contenente una serie di informazioni utili. In testa al file sarà presente un commento che spiega come utilizzare il servizio della libreria che si sta implementando in un'altra applicazione. Suggerisco di leggerlo per comprendere meglio il funzionamento del tutto. Il file Class1.cs conterrà poi un'interfaccia, e due classi.

```
[ServiceContract()]
  public interface IService1
  {
      [OperationContract]
      string MyOperation1(string myValue);
      [OperationContract]
      string MyOperation2(DataContract1
                                       dataContractValue);
  }

  public class service1 : IService1
  {
      public string MyOperation1(string myValue)
      {
          return "Hello: " + myValue;
      }
      public string MyOperation2(DataContract1
                                        dataContractValue)
      {
          return "Hello: " +
```

```
                dataContractValue.FirstName;
        }
    }

    [DataContract]
    public class DataContract1
    {
        string firstName;
        string lastName;

        [DataMember]
        public string FirstName
        {
            get { return firstName; }
            set { firstName = value; }
        }
        [DataMember]
        public string LastName
        {
            get { return lastName; }
            set { lastName = value; }
        }
    }
```

La classe IService1 rappresenta l'interfaccia del servizio, cioè le operazioni che esso esporrà al client, in questo caso MyOperation1 e MyOperation2. L'attributo ServiceContract indica proprio che si tratta di un contratto di servizio WCF.
La classe service1 implementa l'interfaccia precedente.
Infine la classe DataContract1 rappresenta un contratto di dati. L'attributo DataContract indica che la classe implementa un contratto di dati, che potranno dunque essere serializzati e scambiati fra servizio e client.
Naturalmente tali classi e interfacce saranno ora sostituite da quelle necessarie all'implementazione del servizio della pizzeria.
Prima di andare ad implementare il servizio si noti anche la presenza di due using fondamentali in qualsiasi servizio WCF:

```
using System.ServiceModel;
using System.Runtime.Serialization;
```

Il file class1.cs può anche essere eliminato. Si aggiunga poi una nuova classe, chiamandola Pizza. La classe sarà marcata con l'attributo *DataContract*, i membri della classe che saranno esposti dal servizio dovranno essere marcati invece con *DataMember*:

```
[DataContract]
public class Pizza
{
        [DataMember]
        public Guid PizzaId;
        [DataMember]
        public string Name;
        [DataMember]
        public decimal Price;
        [DataMember]
        public string[] Ingredients;
}
```

Per semplicità e comodità si aggiunga un costruttore per inizializzare una pizza in maniera appropriata:

```
public Pizza(string name,decimal price, string[]
                                        ingredients)
{
            PizzaId = Guid.NewGuid();
            Name = name;
            Price = price;
            Ingredients = ingredients;
}
```

Con questo abbiamo creato il contratto strutturale del servizio, cioè i messaggi scambiati fra servizio e client. Adesso si implementerà invece il comportamento, quindi i contratti di behavior, tramite un'interfaccia marcata dall'attributo ServiceContract.
Da Visual Studio 2005 basta fare clic con il destro sul progetto, scegliere Add Item e poi Interface come tipo di elemento da aggiungere. Il nome della nuova interfaccia del servizio sarà IPizzaService, si noti anche la presenza del parametro Namespace per definire uno spazio dei nomi unico per il servizio:

```
using System.ServiceModel;

namespace PizzaWCFServiceLibrary
{
    [ServiceContract(Namespace =
                        "http://IoProgrammo.Services")]
    interface IPizzaService
    {
    }
}
```

A questo punto si può aggiungere una operazione al servizio, per esempio quella per ottenere il listino delle pizze. L'attributo OperationContract indica proprio che si tratta di un'operazione facente parte del contratto del servizio.

```
[OperationContract]
List<Pizza> GetListinoPizze();
```

Non resta che implementare la logica del servizio. Si aggiunga al progetto una classe pubblica PizzaService che implementi l'interfaccia IPizzaService. In questo caso, dato che non fa parte dello scopo dell'articolo, il listino di pizze sarà cabla-

to nel codice, mentre per un'applicazione reale esso sarà magari letto da un database come SQL Server.

```
[ServiceBehavior(InstanceContextMode=InstanceCont
                                    extMode.PerCall)]
public class PizzaService:IPizzaService
{
    private List<Pizza> pizze;
    public PizzaService()
   {
            pizze = new List<Pizza>();
            pizze.Add(new Pizza("margherita", 3.0M,
new string[] { "mozzarella", "pomodoro" }));
            pizze.Add(new Pizza("prosciutto", 3.0M,
new string[] { "mozzarella", "pomodoro", "prosciutto"
                                                }));
            pizze.Add(new Pizza("capricciosa", 3.0M,
new string[] { "mozzarella", "pomodoro",
"prosciutto", "wurstel", "olive", "carciofini","funghi"
                                                }));
    }

    public List<Pizza> GetListinoPizze()
    {
        return pizze;
     }
}
```

Per mezzo dell'attributo applicato alla classe, ed in particolare del parametro InstanceContextMode si è indicato che il servizio verrà istanziato ad ogni chiamata.
Adesso il servizio è pronto per l'uso, non resta che creare un'applicazione client da cui creare un'istanza del servizio e poi configurare il protocollo di comunicazione utilizzato dal servizio stesso. Per farlo si utilizzerà un tool fornito con il nuovo SDK di Windows, il Service Configuration Editor, avviabile sia dal menù Start di Windows, o più comodamente dal menù Tools di Visual Studio 2005.
Dopo aver creato un altro progetto per un'applicazione Console, si aggiunga il file di configurazione dell'applicazione App.config, e lo si apra dal Service Configuration Editor. Dall'albero a sinistra si selezioni il nodo Advanced, poi Service Behavior e si faccia clic su New Service Behavior Configuration dal pannello Task.
Si nomini la configurazione MetadataBehavior nel pannello Behavior e poi si faccia clic sul pulsante Add. Si scelga l'elemento serviceMetadata e lo si aggiunga ancora con Add.
L'elemento creato verrà aggiunto sotto il nodo MetadataBehavior dell'albero a sinistra, lo si selezioni con il mouse per configurarlo e poi per utilizzare il protocollo http si cambi la proprietà HttpGetEnabled a true.
Ora è possibile aggiungere il servizio PizzaService alla configurazione. Nel pannello Tasks si faccia clic sul link Create a New Service, quando viene richiesto il Service Type si faccia clic sul pulsante Browse per cercare l'assembly precedentemente creato, PizzaWCFServiceLibrary.dll, che si troverà nella directory bin del progetto WCF Library. Nel passo successivo sarà richiesta l'interfaccia da utilizzare ed in questo caso l'unica selezionabile sarà la IPizzaService, mentre come tipo di comunicazione si dovrà scegliere http.
Nel passo seguente si lasci selezionato il valore di default, Basic Web Services interoperability, e nell'ultimo passo, come indirizzo del servizio si specifichi:

```
http://localhost:8081/IoProgrammo/PizzaService
```

Nell'albero Configuration a sinistra si noterà ora la presenza del servizio creato, con un singolo EndPoint così come appena configurato (vedi figura 12).

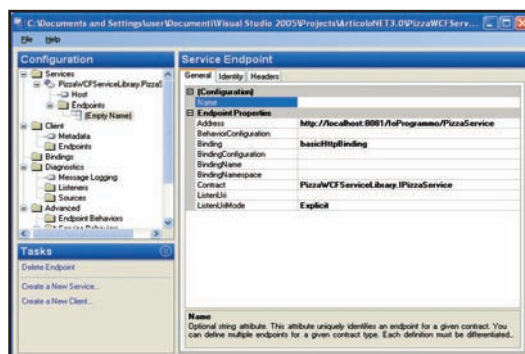



Fig. 12: ***Configurazione di un servizio WCF***

Si selezioni il nodo del servizio PizzaService dall'albero, impostando il parametro Behavior Configuration al valore MetadataBehavior e si salvi il progetto dal menù File, con questo il tool può essere chiuso.
Tornando a Visual Studio il file di configurazione App.config conterrà nuovi elementi con le modifiche appena apportate:

```
<behaviors>
            <serviceBehaviors>
                <behavior name="MetadataBehavior">
                    <serviceMetadata
                                httpGetEnabled="true" />
                </behavior>
            </serviceBehaviors>
</behaviors>
<services>
<service behaviorConfiguration="MetadataBehavior"
name="PizzaWCFServiceLibrary.PizzaService">
            <endpoint
        address="http://localhost:8081/IoProgrammo
                                        /PizzaService"
            binding="basicHttpBinding"
```

```
                        bindingConfiguration=""
contract="PizzaWCFServiceLibrary.IPizzaService" />
</service>
```

Il passo finale è quello di ospitare il servizio in un application domain.
Nel metodo Main dell'applicazione Console viene creato un oggetto ServiceHost che ospiterà il servizio PizzaService, all'indirizzo specificato come parametro.

```
static void Main()
{
        ServiceHost shPizzaService = new ServiceHost(
            typeof(PizzaService), new
            Uri("http://localhost:8081/IoProgrammo/"));
        shPizzaService.Open();
        Console.Write("Premi un tasto per fermare il
                                            servizio");
        Console.Read();
        shPizzaService.Close();
}
```

Il metodo Open mette il servizio in stato di esecuzione e attesa. La chiamata a Console.Read successiva serve a far rimanere il ServiceHost in questo stato, per permettere poi la chiusura alla pressione di un tasto.
Mandando in esecuzione l'applicazione console, si apra un browser digitando l'indirizzo specificato come Uri. Apparirà, se tutto funziona correttamente una pagina come quella mostrata in figura 13, che spiega come creare un client ed utilizzare il servizio.

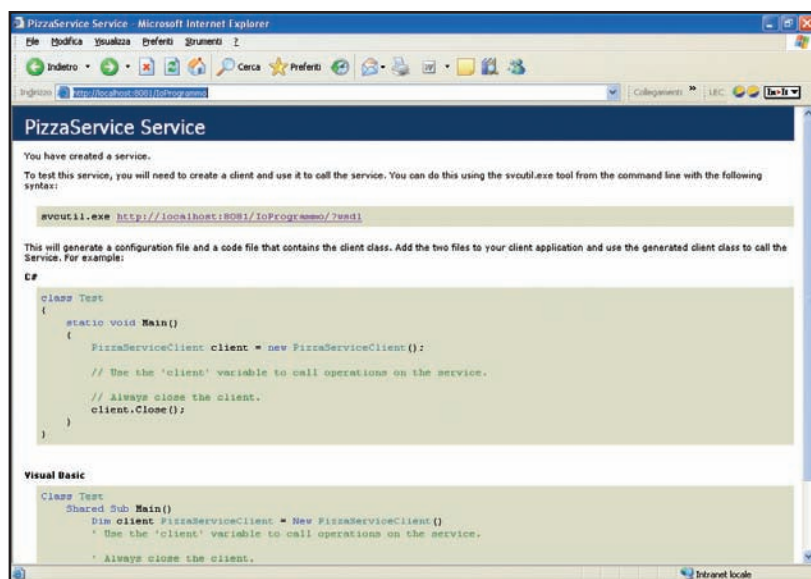



Fig. 13: ***La pagina descrittiva del servizio PizzaService***

Seguendo le istruzioni, si creerà un proxy del servizio, che un'applicazione qualunque potrà utilizzare per comunicare con il servizio vero e proprio.

Innanzitutto è necessaria un'applicazione client, quindi in un'altra soluzione di Visual Studio, si crei un progetto Applicazione Windows Forms, chiamandolo ad esempio PizzaWCFClient, si crei il file di configurazione App.config e come dice la pagina di help, si lanci da un prompt dei comandi di Visual Studio il comando seguente, mentre il ServiceHost è ancora attivo, spostandosi nella cartella del nuovo progetto, contenente il file App.config:

```
svcutil http://localhost:8081/IoProgrammo
      /out:PizzaServiceProxy.cs /config:app.config
                                    /mergeconfig
```

che creerà un file PizzaServiceProxy.cs ed aggiornerà il file app.config esistente. Si aggiunga il file PizzaProxyService.cs al progetto, e si verifichi che il file di configurazione sia stato aggiornato.
Ora è possibile utilizzare la classe PizzaServiceClient in maniera piuttosto immediata, ottenere il listino delle pizze e magari mostrarlo in un controllo TreeView.

```
PizzaServiceClient client = new PizzaServiceClient();
Pizza[] pizze=client.GetListinoPizze();
```

La figura 14 mostra l'applicazione client in esecuzione, il cui codice è naturalmente allegato all'articolo e può essere quindi provato sul proprio computer.

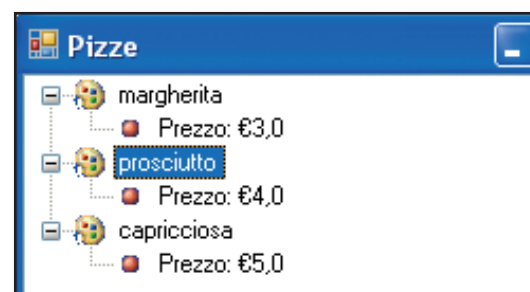



Fig. 14: ***L'applicazione client del servizio PizzaService***

IL MODELLO APPLICATIVO DI WPF

Verrà ora affrontato lo sviluppo di un'applicazione WPF, cercando di dare una panoramica la più ampia possibile della nuova tecnologia di costruzione delle interfacce grafiche.
Il modello applicativo di Windows Presentation Foundation distingue fra due tipi di applicazioni: standalone e browser. Un'applicazione del primo tipo mostra il contenuto grafico nelle classiche finestre, dialog, e message box, mentre un'applicazione browser è fatta da pagine mostrate in un browser (attenzione a non confonderle con le applicazioni asp.net). Di conseguenza, le applicazioni WPF possono avere due stili di navigazione: tramite menu o

hyperlink. Non è necessario specificare che le applicazioni standalone supportano di default la navigazione a menù, mentre le applicazioni browser utilizzano i collegamenti ipertestuali. Il bello però è che WPF consente un approccio misto.

INTRODUZIONE A XAML

XAML è stato creato da Microsoft per interfacciare il .NET Framework con il sistema operativo Vista per mezzo del substrato di presentazione WPF. XAML da agli sviluppatori l'opportunità di controllare il layout di tutti gli elementi di un'interfaccia come caselle di testo, pulsanti, immagini e altro ancora, utilizzando una sorta di dialetto XML.
Dato che ogni elemento XAML corrisponde ad una classe del CLR, tutto ciò che può essere fatto tramite XAML, può essere fatto via codice.
Anche gli eventi ed i relativi gestori possono essere dichiarati come attributi XAML e poi implementati in C# o VB.net.
XAML offre diversi benefici sia rispetto ad altri linguaggi di definizione delle interface, come XUL e HTML, ma anche rispetto alla tradizionale programmazione in linguaggi di alto livello. Per esempio, per creare un pulsante in XAML con relative gestore dell'evento Click, impostando colore di background e testo si scriverà:

```
<Button Click="OnClickHandler" Background="Green"
                                   Content="Invia" />
```

Per fare la stessa cosa in C# si dovrà scrivere:

```
Button myBtn = new Button(  );
myBtn.Background = Brushes.Green;
myBtn.Text=" Invia ";
myBtn.Click += new
                  System.EventHandler(OnClickHandler);
```

XAML non richiede altro che un editor di testo per essere scritto, ma naturalmente un IDE con intellisense e magari con designer visuale è molto più produttivo. Le estensioni di Visual Studio per WPF consentono di scrivere codice XAML dall'IDE e di progettare le interfacce in maniera visuale.

APPLICAZIONI STANDALONE

Per creare un'applicazione standalone con Visual Studio 2005, dotato delle estensioni WPF, si crei un progetto di tipo Windows Application, dando un nome a propria scelta, per esempio WinAppWPF.
Visual Studio 2005, creerà due file XAML, App.xaml e Window1.xaml, con i relativi file di code-behind, per esempio App.xaml.cs e Window1.xaml.cs. Il primo è il file che definisce l'intera applicazione.
Ogni applicazione WPF infatti è un'istanza di System.Windows.Application, che può essere usata appunto da XAML, da codice o da una loro combinazione.
L'applicazione appena creata da Visual Studio conterrà il seguente XAML:

```
<Application x:Class="WinAppWPF.App"
xmlns="http://schemas.microsoft.com/winfx/2006/xa
                                    ml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/
                                               xaml"
    StartupUri="Window1.xaml"
>
    <Application.Resources>
    </Application.Resources>
</Application>
Mentre dal lato codice si avrà:
public partial class App : System.Windows.Application
{
}
```

Ma dove si trova il metodo Main? Non c'è, perché il punto di ingresso dell'applicazione è incapsulato dalla classe Application, e la proprietà StartupUri indica quale finestra utilizzare come interfaccia iniziale dell'applicazione, in questo caso Window1.xaml. Il fatto che il file App.xaml e App.xaml.cs costituiscono il punto di partenza viene indicato a VS e quindi al compilatore tramite la finestra delle proprietà di VS. Come si può notare in figura 15, dopo aver selezionato il file App.xaml, la proprietà Build Action è impostata ad ApplicationDefinition. Ciò farà in modo che dietro le quinte verrà generato il codice C# seguente:

```
public static void Main() {
           WinAppWPF.App app = new
                                WinAppWPF.App();
           app.InitializeComponent();
           app.Run();
}
```

Ma tutto ciò può tranquillamente essere ignorato dallo sviluppatore.
L'altro file generato, Window1.xaml, contiene la definizione di una finestra, istanza in WPF della classe Window.
La proprietà Build Action per questo file è stavolta impostato al valore Page, e la finestra quindi può essere utilizzata dall'Application come finestra principale per mezzo del suo Uri, come visto in precedenza. Visual Studio 2005 consente di editare manualmente il codice XAML, oppure di creare e modificare l'interfaccia grafica per mezzo del designer (vedi figura 15)

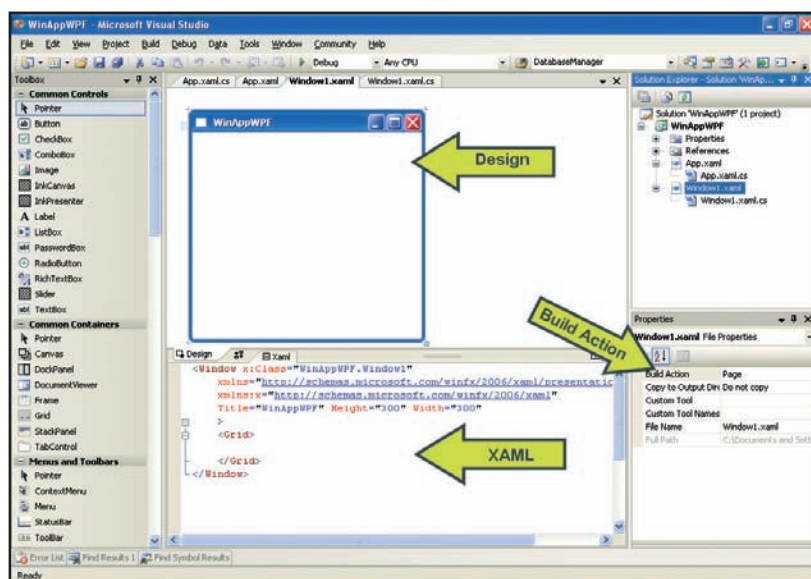



Fig. 15: ***Il designer di Visual Studio per WPF***

Tramite il designer si possono utilizzare i controlli della ToolBox come ormai tutti gli sviluppatori Windows dovrebbero essere abituati a fare, trascinandoli su un contenitore della finestra.
Il codice XAML della finestra generata da Visual Studio assomiglierà al seguente:

```
<Window x:Class="WinAppWPF.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xa
                                    ml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/
                                               xaml"
    Title="WinAppWPF" Height="300"
                      Width="300"
                                  xmlns:my="clr-
            namespace:System;assembly=mscorlib"
    >
  <Grid>
  </Grid>
</Window>
```

Ogni finestra deve avere un contenitore in cui posizionare gli elementi di interfaccia. Microsoft, come si può notare, incoraggia l'utilizzo della classe Grid, per la flessibilità di posizionamento.
Per aggiungere un pulsante alla finestra basta creare all'interno dell'elemento Grid un sottoelemento Button, per esempio:

```
<Grid>
        <Button Background="AliceBlue"
                Foreground="Black"
                Width="100"
                Height="20"
                Content="Submit"
                Click="Button_Click" />
  </Grid>
```

Nel file di code-behind si andrà poi a implementare il gestore Button_Click:

```
protected void Button_Click(object sender,
                            RoutedEventArgs ev)
{
MessageBox.Show("Ciao");
}
```

APPLICAZIONI PER BROWSER

Per creare un'applicazione browser, dall'elenco dei template di Visual Studio 2005 si scelga la tipologia XAML Browser Application.
VS 2005 creerà ancora una volta una coppia di file App.xaml e App.xaml.cs per l'applicazione, ma stavolta, piuttosto che una Window, creerà una Page, in quanto le applicazioni per browser saranno costituite da pagine piuttosto che da finestre. Il funzionamento di una Page è praticamente identico ad una Window, quindi è possibile aggiungervi un Button ed una Label come di seguito:

```
<Page x:Class="XAMLBrowserApplication.Page1"
xmlns="http://schemas.microsoft.com/winfx/2006/xa
                                    ml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/
                                               xaml"
    Title="Page1"
    >
    <Grid>
<Button Name="button1" Click="Button1_Click"
                Height="27" Margin="0,0,8,8"
                VerticalAlignment="Bottom"
        HorizontalAlignment="Right" Width="100">
Vai a pagina 2
</Button>
        <Label Height="25" Margin="120,75,145,0"
          Name="label1" VerticalAlignment="Top">
Contenuto della pagina 1
</Label>
  </Grid>
</Page>
```

Si aggiunga poi un'altra pagina, facendo clic con il destro sul progetto, scegliendo Add dal menù contestuale, e selezionando il tipo Page dall'elenco. Per spostarsi dalla Page1 alla Page2 come in una normale applicazione web, si utilizza la classe NavigationService ed il suo metodo Navigate.

```
protected void Button1_Click(object sender,
                            RoutedEventArgs ev)
{
      NavigationService.Navigate("page2.xaml");
}
```

Oppure, utilizzando un controllo Hyperlink, basta impostare la proprietà NavigateUri:

```
<Hyperlink NavigateUri="page2.xaml">Vai a pagina
                                   2</Hyperlink>
```

Avviando l'applicazione (a proposito, per abilitare il debug di una applicazione per browser è necessario, nelle proprietà del progetto, sezione Debug, selezionare la voce Enable Unmanaged Code Debugging) si aprirà il browser Internet, con l'applicazione che verrà eseguita al suo interno, come mostrato in figura 16.
Una bella novità introdotta con WPF, è la possibilità di creare applicazioni ibride, sfruttando il concetto di navigazione anche in una normale applicazione. Basta utilizzare un oggetto NavigationWindow:

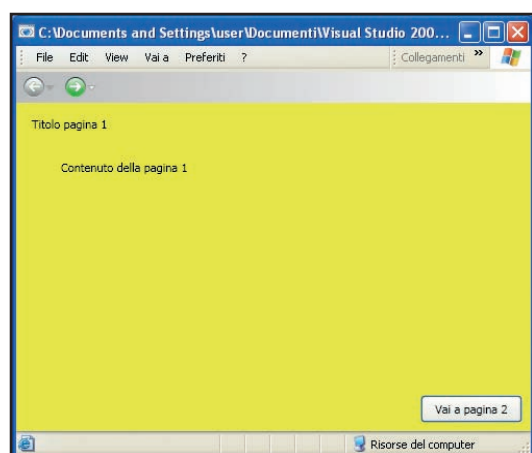



Fig. 16: ***Una browser application in esecuzione***

```
<NavigationWindow ... Source="Page1.xaml" />
```

Una possibile applicazione reale di Navigation Window potrebbe essere la realizzazione di un Wizard.

WINDOWS FORMS IN WPF

Che fine faranno i controlli Windows Forms che ogni sviluppatore è abituato a utilizzare, o quelli custom che sono costati giorni o mesi? Tranquilli, possono essere tranquillamente utilizzati in WPF.
Esiste la classe WindowsFormsHost il cui scopo è proprio quello di contenere un qualunque controllo Windows Forms, standard o custom.
Per utilizzare tale classe è necessario aggiungere al progetto WPF i due riferimenti agli assembly System.Windows.Forms.Integration e System.Windows.Forms. Come esempio riutilizziamo quanto visto nel paragrafo su WCF, in particolare il controllo TreeView, contenente le pizze ricavate dal servizio PizzaService.
Se la finestra Window1 è così definita in XAML:

```
<Window x:Class="WPF_Hosts_Winforms.Window1"
xmlns="http://schemas.microsoft.com/winfx/2006/xa
                                    ml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/
                                               xaml"
    Title="WPF_Hosts_Winforms" Height="300"
                                        Width="300"
    Loaded="Window1_Loaded"
  >
  <Grid Name="grid1">
  </Grid>
</Window>
```

Si può creare da codice l'istanza host di WindowsFormsHost ed aggiungerla alla Grid denominata grid1 così:

```
void Window1_Loaded(object sender,
                                RoutedEventArgs e)
{
            WindowsFormsHost host = new
                                WindowsFormsHost();
            System.Windows.Forms.TreeView treeView1
                                                  =
new System.Windows.Forms.TreeView();
            PopulateTree(treeView1);
            host.Child
          =(System.Windows.Forms.Control) treeView1;
            this.grid1.Children.Add(host);
}
```

Il metodo PopulateTree, che si tralascia rimandando al paragrafo Creare servizi e client WCF, si occupa di ricavare dal proxy del servizio PizzaService il listino delle pizze e popolare l'albero.

CONCLUSIONI

Nell'articolo si è data un'introduzione di .NET Framework 3.0, e dei suoi componenti fondamentali, dedicati alla comunicazione, alla creazione di applicazioni basate su workflow, alla definizione di moderne interfacce grafiche, ed alla gestione delle identità digitale. Si è cercato di creare qualche esempio introduttivo per mostrare il nuovo modo di sviluppare che attende alla porta gli sviluppatori Windows, ma certamente sarà necessario tornare sui singoli argomenti, cosa che faremo sicuramente nei prossimi numeri

Antonio Pelleriti

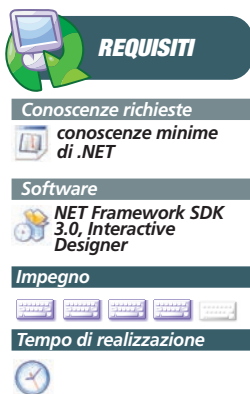
GLI STRUMENTI DI SVILUPPO DEL FUTURO

EXPRESSION: È QUESTO IL NOME CHE MICROSOFT HA VOLUTO DARE ALLA SUA NUOVA SUITE PER LA CREAZIONE DI INTERFACCE GRAFICHE E CONTENUTI MULTIMEDIALI. LA NOVITÀ È CHE QUESTA VOLTA TUTTO È PENSATO PER AIUTARE GLI SVILUPPATORI...

Conoscenze richieste
conoscenze minime di .NET

Software
NET Framework SDK 3.0, Interactive Designer

Impegno

Tempo di realizzazione

Microsoft Expression è una nuova suite di prodotti pensata per sviluppatori e designer operanti sia sul web sia sulle applicazioni desktop.
0dotti: Interactive Designer, è il nuovo strumento per sviluppatori dedicato alla progettazione di interfacce grafiche secondo una concezione innovativa ed in grado di renderle molto più accattivanti delle GUI attuali.
Web Designer è praticamente il successore di FrontPage e quindi è dedicato allo sviluppo web affiancandosi a Visual Studio 2005, data la natura ormai dinamica della maggioranza dei siti web.
Graphic Designer: è infine la nuova applicazione di grafica e manipolazione delle immagini, destinato a competere con mostri sacri quali possono essere Photoshop e Illustrator.
Per ognuno dei prodotti della suite Expression sono disponibili al momento delle CTP (Community Technology Preview), liberamente scaricabili dal sito ufficiale di Microsoft Expression (vedi box laterale), e che permettono di conoscere e familiarizzare con i prodotti, prima dell'uscita della versione definitiva. Naturalmente è sconsigliato utilizzare le CTP in fase di produzione e con dati sensibili, in quanto potrebbero esserci bachi o malfunzionamenti non ancora scovati. Scopo di questo articolo è fornire una panoramica di insieme della suite Microsoft Expression. Focalizzeremo poi la nostra attenzione su Interactive Designer che è uno strumento che potrebbe aiutare moltissimo il programmatore nello sviluppo di interfacce grafiche

INSTALLAZIONE DEGLI EXPRESSION TOOLS

Per utilizzare i tools Expression, è necessario installare o possedere una versione del .NET Framework. Per Expression Web Designer è sufficiente la versione 2.0, mentre per Interactive Designer e Graphic Designer è necessario aggiornare alla versione 3.0 di .NET. I sistemi operativi supportati sono Windows XP con almeno SP2, o successivi. Windows Vista non ha bisogno di altre installazioni, in quanto .NET Framework 3.0 è parte integrante del sistema operativo.
Una volta scaricati i file di setup dei tre tool o di quelli che desiderate utilizzare, l'installazione non richiede nulla di particolare rispetto ad un qualunque altro applicativo Windows, lanciate i file setup.exe ed attendete il termine della procedura guidata.

LE NUOVE TECNOLOGIE

Alla base dei tool della serie Expression sta la tecnologia WPF (Windows Presentation Foundation), che a sua volta è anche un componente fondamentale del nuovo .NET Framework 3.0 e dunque di Windows Vista.
WPF, conosciuto in precedenza anche come Avalon, è il sistema unificato di presentazione che Microsoft utilizzerà per le future versioni di Windows. Il suo motore si avvantaggia dell'hardware di ultima generazione e fornisce classi che gli sviluppatori potranno utilizzare per creare applicazioni ricche di animazioni e contenuti multimediali.
Tutto ciò è basato sull'altrettanto nuova tecnologia chiamata XAML (Extensible Application Markup Language), un linguaggio di markup che permette di definire e specificare il comportamento e l'aspetto dell'interfaccia utente per mezzo di XML.
Tutti gli strumenti Expression contengono la funzionalità di esportazione in formato XAML, e permettono quindi di sfruttare al massimo le caratteristiche della tecnologia WPF, in maniera semplice e immediata e di facile integrazione con il resto del sistema.. Per esempio sarà possibile disegnare pulsanti dalla grafica accattivante, esportarli in XAML, ed utilizzarli in una qualunque applicazione Windows, senza essere più legati ai classici e grigi pulsanti a cui siamo abituati da decenni. Vediamoli nel dettaglio.

INTERACTIVE DESIGNER

Expression Interactive Designer è lo strumento per la creazione di interfacce e contenuti grafici per la piattaforma Windows. Interactive Designer contiene una completa serie di strumenti di disegno vettoriale e bitmap, per la creazione di animazioni, di grafica 3D ed integrazione con altri contenuti multimediali. La cosa probabilmente più interessante per programmatori e sviluppatori, abituati a masticare codice è la possibilità di passare dalla vista per il design a quella che mostra il corrispondente codice XAML, in pratica in maniera totalmente analoga a quanto si è abituati a fare in Visual Studio, sia in ambito Windows, che in ambito Web con le pagine aspx suddivise in parte grafica e parte HTML.

Come si può notare dalla figura al lato, la stessa interfaccia grafica di Interactive Designer è basata su WPF, per consentire un livello superiore di controllo del layout e dei singoli elementi di una GUI.

Si vedrà ora come utilizzare Interactive Designer per realizzare la nostra prima applicazione che sfrutti i prodotti Expression, magari esplorando qualche caratteristica più avanzata. Come quasi sempre accade, il nostro progetto implementerà l'immancabile applicazione per la ricerda degli arretrati dei numeri di ioProgrammo.

Il primo passo è aprire Interactive Designer e creare un nuovo progetto. Basta selezionare il menù File e poi New Project. A questo punto si potrà scegliere la tipologia di progetto fra due possibili, Standard Application o Control Library. Si selezioni la prima e si dia un nome al progetto, per esempio ioProgrammo, come si può vedere in **figura 3**:

Sempre nella stessa finestra di dialogo è possibile scegliere il linguaggio di sviluppo, fra C# e Visual Basic, in questo caso lasciamo la scelta su C#.

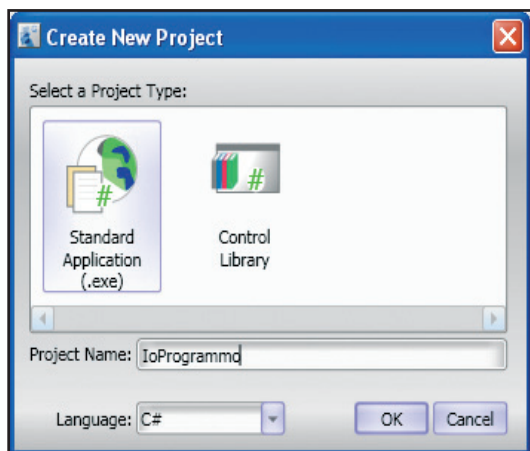



Fig. 3: *La selezione del tipo di progetto*

Dopo aver fatto clic su OK si aprirà una nuova scena, vuota, chiamata di default Scene1.xaml, e che costituirà la nostra superficie di lavoro. È possibile rinominare la scena e dunque i relativi file a proprio piacimento, basta esplorare sulla destra la finestra Projects, selezionare il file Scene1.xaml e fare clic con il tasto destro per poter scegliere il comando Rename.

Chiamiamolo ioProgrammo.xaml e notiamo come anche il file .cs sottostante, visualizzabile espandendo l'albero, verrà automaticamente rinominato in ioProgrammo.xaml.cs.

Dalla stessa interfaccia di Interactive

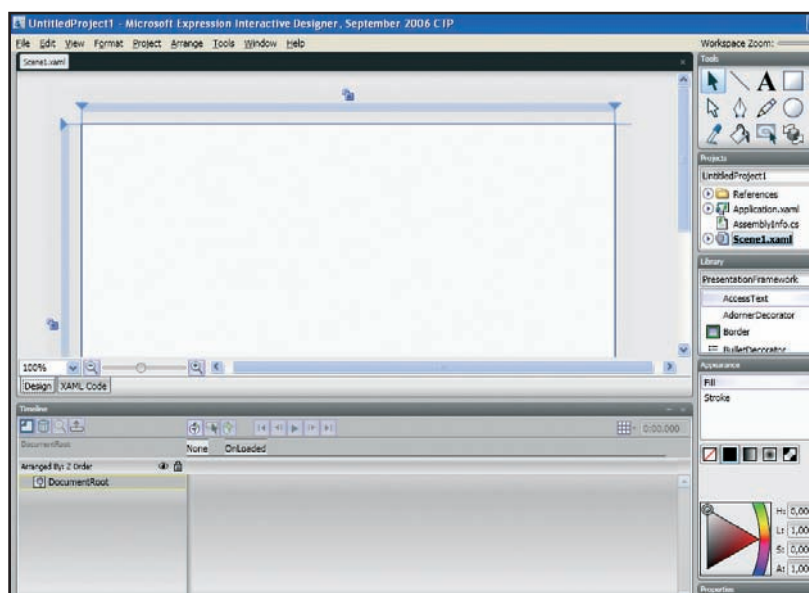



Fig. 4: *L'interfaccia grafica di Interactive Designer*

GRAPHIC DESIGNER

Expression Graphic Designer è lo strumento per la creazione di grafica vettoriale o bitmap. La figura seguente (figura 1) mostra l'interfaccia del tool,

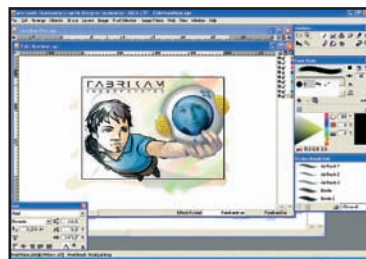



Fig. 1: *Un'immagine creata con Graphic Designer.*

con uno dei disegni facenti parte del tutorial e che rende bene l'idea delle possibilità di Graphic Designer.
Essendo pienamente integrato nella suite e predisposto a supportare gli strumenti di sviluppo futuri ed il .NET Framework 3.0 è ovviamente dotato della possibilità di esportare un qualunque disegno in formato XAML.
Come già detto questo formato è facilmente utilizzabile per la definizione di interfacce grafiche e contenuti avanzati di Windows Presentation Foundation.

Designer è possibile visualizzare e modificare il codice C#, sfruttando anche le funzionalità di evidenziazione della sintassi e di Intellisense tipiche di Visual Studio o degli IDE moderni.
Per inserire un qualsiasi controllo, basta selezionarlo dalla finestra Library e poi disegnarlo sulla scena. Se la palette Library non fosse visibile basta selezionarla dal menu View.
Supponendo di voler creare un'applicazione per consultare i numeri arretrati di ioProgrammo, utilizzeremo una ListBox che conterrà l'elenco, una casella di testo che poi conterrà il sommario del numero selezionato ed un oggetto Image che visualizzerà la copertina del numero selezionato.
L'interfaccia così pensata apparirà all'interno di Interactive Designer come nella figura seguente:
In questo caso si è utilizzato come contenitore principale un oggetto Grid, all'interno del quale si sono inseriti i controlli ListBox, TextBox e Image suddetti. L'oggetto grid ci servirà per avere un componente comune da utilizzare per il DataBinding, come si vedrà nel seguente paragrafo.

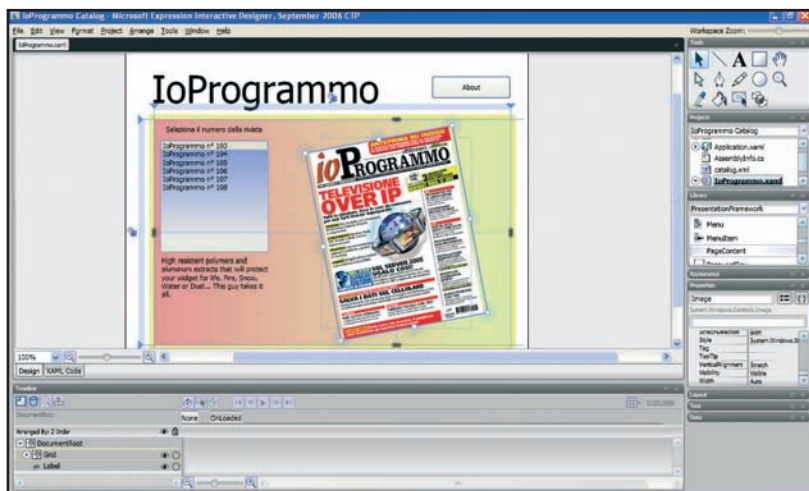



Fig. 5: *Il progetto dell'interfaccia dell'applicazione*

SORGENTI DI DATI

I dati delle riviste da visualizzare saranno contenuti e letti da un file XML che avremo precedentemente preparato Interactive Designer permette di aggiungere una sorgente dati XML semplicemente con pochi clic del mouse. Innanzitutto creiamo un file con una struttura adatta al nostro esempio, il seguente è un estratto di tale XML:

```
<Catalog>
```

```
    <Products>
        <Rivista>
            <Name>IoProgrammo
                n° 103</Name>
            <Image>Images\4-
                103g.jpg</Image>
            <Description>
                Sommario
                del numero
            </Description>
            <Price>€5.90</Price>
        </Rivista>
...
</Products>
</Catalog>
```

A questo punto il file XML può essere utilizzato da Interactive Designer per ricavare i dati da mostrare nell'applicazione. Bisogna innanzitutto aggiungere un nuovo XML Data Source. Basta visualizzare la finestra Data, quindi fare clic su Add XML Data Source e poi dalla finestra di selezione del file (vedi figura 6) sfogliare per trovare il file XML appena creato.

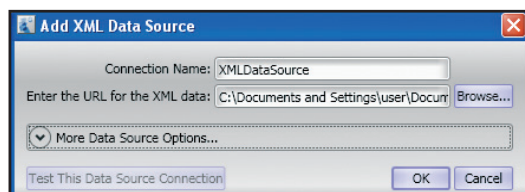



Fig. 6: *Aggiungere una nuova sorgente dati*

Ora possiamo procedere al binding dei dati. Innanzitutto selezioniamo la proprietà DataContext dell'oggetto Grid. Il DataContext consente agli elementi figli di un oggetto,

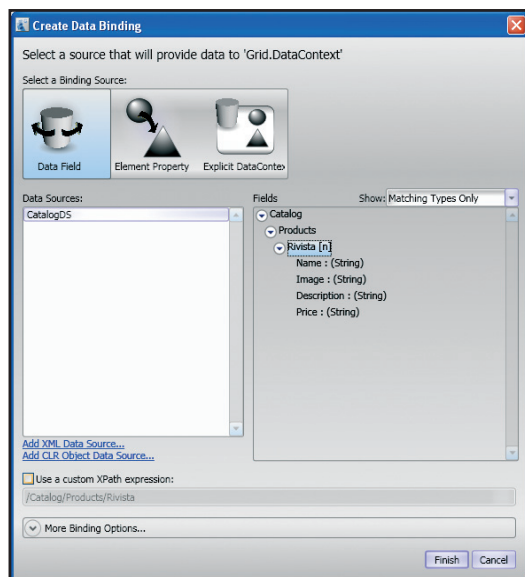



Fig. 7: *Connessione alla sorgente dati*

come ListBox e Image, di ereditare informazioni sulle sorgenti di dati dai propri elementi padri, in questo caso Grid.
Cliccando sulla proprietà DataContext apparirà il menù contestuale, e da questo si potrà selezionare la voce Databind.
Dalla finestra di dialogo aperta a questo punto aperta (mostrata in figura 7), potremo selezionare l'elemento Rivista dall'albero della nostra sorgente dati e poi fare clic su Finish. Ora sarà possibile utilizzare le informazioni ricavata dal file XML e visualizzarle nei diversi componenti dell'interfaccia grafica, il tutto all'interno di Interactive Designer.
Dopo aver aggiunto una ListBox alla Grid, e magari dopo averne modificato l'aspetto a proprio piacimento con gli strumenti grafici a disposizione, si faccia clic con il tasto destro su di essa, selezionando la voce Bind to Data dal menù contestuale.
Nella finestra di dialogo per il Data Binding questa volta si selezioni prima Explicit DataContext come Binding Source, in quanto vogliamo utilizzare la sorgente precedentemente impostata per la Grid.
In questo caso vogliamo che la ListBox mostri solo il numero della rivista, cioè solo l'elemento Name letto dal file XML. Quello che si fa in questo caso è creare un template ad hoc ed applicarlo al controllo ListBox. Il tutto può essere fatto da interfaccia grafica oppure da codice XAML, per esempio:

```
<DataTemplate x:Key="RivistaTemplate1">
    <StackPanel x:Name="StackPanel">
        <TextBlockx:Name="TextBlock"
Text="{Binding Mode=OneWay, XPath=Name}"/>
    </StackPanel>
</DataTemplate>
Ed applicandolo poi impostando la proprietà
ItemTemplate della ListBox:
<ListBox x:Name="ListBox"
    ItemTemplate="{DynamicResource
        RivistaTemplate1}">
```

Un'altra possibilità, forse più immediata, è quella di creare la ListBox ed il suo template visivamente a partire dalla sorgente dati.
Dalla finestra Data selezioniamo il nodo Rivista[n] e trascinimolo sulla Grid, rilasciando il mouse apparirà un menù contestuale che permette di scegliere la tipologia di controllo da utilizzare per rappresentare i dati. Scegliamo una ListBox in questo caso ed il campo ItemsSource come destinazione. Subito dopo potremo creare il template da utilizzare. Si selezioni solo il campo Name in quanto solo questo sarà visualizzato nella lista.
Allo stesso modo si può creare un'immagine bindata all'elemento Image della Data Source. In questo caso però bisogna prima aggiungere un controllo Image alla Grid, poi selezionare la sua proprietà Source e fare clic su Databind. Ancora una volta come Binding Source si scelga Explicit DataContext e stavolta come campo si scelga Image.
Tralasciamo la procedura per aggiungere un campo di testo e collegarlo all'elemento Description che conterrà il sommario della rivista, in quanto perfettamente analogo,

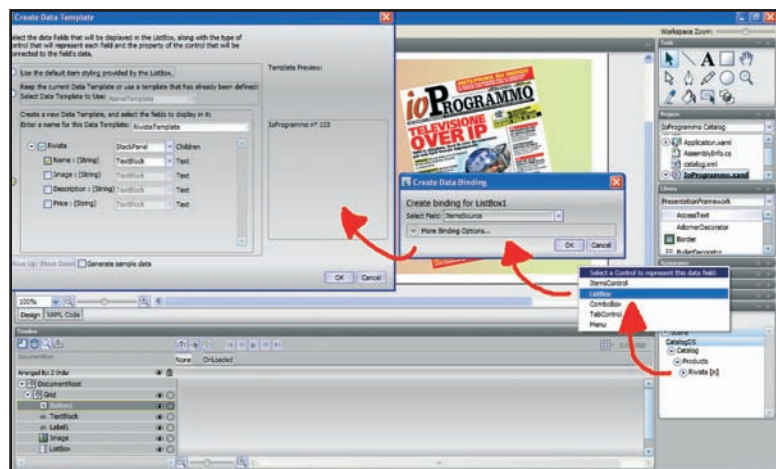



Fig. 8: ***Creare un template per la ListBox***

WEB DESIGNER

Expression Web Designer è lo strumento per la creazione è manutenzione di siti web statici o dinamici in asp.net, e basati su standard. La figura 2 mostra l'interfaccia del tool:

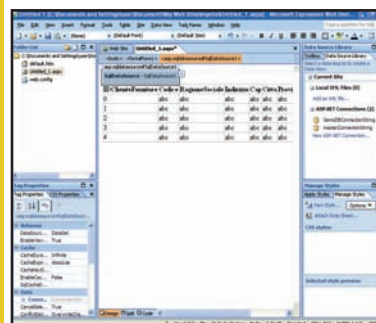



Fig. 2: ***Web Designer all'opera.***

Web Designer permette in maniera semplice di verificare la conformità agli standard, consentendo quindi di apportare le necessarie correzioni in fase di sviluppo e non dopo aver messo on line il sito e dopo averlo validato. È inoltre possibile generare stili ed applicarli agli elementi tramite la sua interfaccia in maniera notevolmente migliorata rispetto alle scarne possibilità dell'editor HTML di Visual Studio 2005.
Allo stesso modo è possibile creare layout professionali e WISYWIG basati su CSS. Così come in Visual Studio è poi possibile creare siti che necessitano di accesso ai dati, per esempio utilizzando i controlli DataSource di ASP.NET, per connettersi a database Access, SQL Server, Oracle.
Citiamo infine, essendo un prodotto Microsoft, il pieno supporto per le caratteristiche avanzate di ASP.NET 2.0, vale a dire Data Binding, nuovi controlli, pagine Master, stili, e debugging mediante server di sviluppo locale.

basta effettuare il Data Binding sulla proprietà Text.

TEST DELL'APPLICAZIONE

Chi è più curioso potrà visualizzare a questo punto il codice XAML prodotto da Interactive Designer, semplicemente facendo clic sulla linguetta XAML dell'editor.
Soddisfatta tale curiosità si potrà infine eseguire la nostra nuova applicazione. È possibile compilare il progetto dal menù Project per mezzo del comando Build Project e poi testarlo mediante il comando Test Project oppure semplicemente premendo il tasto F5.
Compilato ed eseguito il progetto, il suo aspetto dovrebbe essere simile a quello della figura seguente:

MODIFICARE IL CODICE

Noterete nell'applicazione di esempio, la presenza di un pulsante Button, aggiunto anch'esso tramite l'interfaccia di Interactive Designer ed il cui codice XAML è il seguente:

```
<Button HorizontalAlignment="Right"
    VerticalAlignment="Top" Margin="0,-69,0,0"
    Width="126" Height="36" x:Name="Button1"
    Content="About"/>
```

Fig. 9: *L'applicazione in esecuzione*

Per gestire il click su di esso apriamo a questo punto il progetto con Visual C# Express Edition. I più esperti di C# saranno in grado di gestire l'evento Click in pochi secondi, semplicemente scrivendo qualche riga di codice all'interno della classe Scene1:

```
public partial class Scene1
{
        public Scene1()
{
                this.InitializeComponent();
                Button1.Click += new
            RoutedEventHandler(Button1_Click);
        }

void Button1_Click(object sender,
                            RoutedEventArgse)
{
        MessageBox.Show("Aplicazione di
        esempio creata con Interactive Designer",
            "About", MessageBoxButton.OK,
                MessageBoxImage.Information) ;
        }
}
```

Ciò mostra come gli strumenti Expression, in questo caso Interactive Designer, siano pienamente integrati e compatibili con Visual Studio 2005.

CONCLUSIONI

Gli strumenti Expression di Microsoft rappresentano un'ottima opportunità per scatenare la propria fantasia e creare applicazioni ricche di contenuti grafici in maniera rapida e veloce. Inoltre essi si integrano perfettamente fra di loro e con Visual Studio 2005, permettendo di lavorare ai diversi aspetti di un'applicazione, da quello puramente grafico al codice .NET sottostante. Nell'articolo si è dato più risalto ad Interactive Designer, gli sviluppatori Web sicuramente saranno più attratti da Web Designer, mentre i grafici da Graphic Designer. Ciò che è importante capire è che XAML si configura come una sorta di formato dati universale utilizzato per creare template a qualunque livello. Si può partire da un'interfaccia in Photoshop importata poi in uno degli strumenti Expressioni, oppure direttamente in Visual Studio, ciò che conta è che lo sviluppatore non è più l'elemento di disomogeneità all'interno di un team di sviluppo, piuttosto ne diventa il collante

Antonio Pelleriti

UNA MAILING LIST GESTITA SUL WEB

ABBIAMO REALIZZATO UN SERVIZIO CHE GESTISCE UNA LISTA DI DISTRIBUZIONE DELLA POSTA SENZA AVERE INSTALLATO NESSUN SOFTWARE LATO SERVER MA SOLO DELLE PAGINE WEB. VEDIAMO ORA COME CREARE IL CLIENT PER LA CONSULTAZIONE

In un precedente articolo abbiamo parlato di come implementare una mailing list attraverso dei Web Services. Avevamo illustrato la costruzione dell'applicazione server, resta quindi da vedere come implementare il Client. Poichè stiamo parlando di un un servizio web le possibilità che abbiamo per realizzare il client sono due: Applicazione desktop, Applicazione Web. Per comodità di distribuzione scegliamo la seconda, l'applicazione Web, ma anche qui ci troviamo di fronte alla scelta di due soluzioni: Applicazione ASP.NET tradizionale, Applicazione AJAX. L'applicazione AJAX consente una maggiore interazione tra client e server, evita il continuo ricaricamento della pagina ed il lavoro aggiuntivo di elaborazione HTML richiesto al server; per contro questa tecnologia non è adatta a tutti i browser in circolazione (o a quelli che ad esempio hanno disabilitato alcune funzioni come javascript). Qui, tuttavia, parliamo di amministrare la Mailing List, operazione che deve essere fatta da una ristretta cerchia di persone a cui possiamo richiedere requisiti software precisi, quindi la nostra scelta si indirizzerà sull'applicazione AJAX. È chiaro che se volessimo invece approntare alcune pagine pubbliche, ad esempio per l'iscrizione e la cancellazione, dovremmo, per queste parti, usare ASP.NET.

mente le liste di messaggi e invia le e-mail.
La struttura dei dati, serializzata in un file xml sul server, è suddivisa in:

- Categorie
- Utenti
- Messaggi

Ogni oggetto Utente contiene una lista di categorie associate. Anche ogni oggetto Messaggio contiene il riferimento ad una lista di categorie, in modo che, in fase di invio, si potrà recuperare la lista di utenti a cui effettuare l'inoltro tra tutti gli utenti che hanno le categorie del messaggio nella propria lista di categorie. Il Messaggio ha anche un Registro associato dove il programma può scrivere l'esito di ogni tentativo di invio, in modo da evitare di inviare più volte lo stesso messaggio ad un utente. La libreria contiene anche una classe, Manager, che fa da interfaccia alle chiamate dei Web Services e consente di manipolare gli oggetti e controllare la serializzazione-deserializzazione dei dati. Per i Web Services è stato progettato, utilizzando i generics, una classe OperationResult che ingloba l'oggetto risposta in una struttura omogenea che contiene anche informazioni sull'esito, eccezioni ecc…

REQUISITI

Conoscenze richieste
conoscenza di base di .NET discreta conoscenza di Javascript, XML e HTML

Software
Nessuno

Impegno

Tempo di realizzazione

LO SCHEMA DEL SERVIZIO

Ricordiamo, in sintesi, le caratteristiche salienti del servizio di mailing list illustrato nella prima parte.
Il servizio non è un servizio di Windows, siamo partiti dal presupposto che dovesse essere possibile far funzionare la Mailing List presso un Internet Provider che offra spazio in Hosting (dove ovviamente non avremmo possibilità di installare niente), quindi, sfruttando gestori di eventi a livello di applicazione posti nel file global.asax (che, lo ricordiamo, non funziona solo per ASP.NET ma anche per i Web Services) facciamo partire un Thread secondario in background che controlla ciclica-

IMPLEMENTAZIONE DEL CLIENT AJAX

Si tratta adesso di costruire l'interfaccia AJAX per il Client ai nostri Web Services. AJAX, lo ricordiamo per l'ennesima volta, non è un linguaggio di programmazione ma significa semplicemente manipolare l'XML con javascript per interagire con la pagina Web. Quello che ci serve quindi è:

1) Un server che ci dia risposte in XML
2) Un set di funzioni Javascript per inviare richieste e ricevere risposte dal server
3) Alcune pagine web HTML dove trovano vita gli scripts.

IL SERVER

Costruendo i Web Services il primo requisito l'abbiamo già soddisfatto, infatti la chiamata a un Web Service dà sempre luogo a una risposta XML (sempre che l'oggetto della risposta sia serializzabile).
Gli unici accorgimenti per utilizzare l'output XML di un Web Service .NET con javascript sono:

1) abilitare il protocollo POST per l'interrogazione al Web Service inserendo nel file web.config del sito dove risiede il servizio con:

```
<configuration>
<system.web>
…
<webServices>
        <protocols>
                <add name="HttpPost"/>
        </protocols>
</webServices>
</system.web>
</configuration>
```

questo ci consentirà di indirizzare chiamate HTTPRequest da Javascript con il metodo POST (non che con SOAP non sia possibile, ma richiederebbe un mucchio di lavoro in più).

2) dichiarare, nella classe del Web Service, un Namespace vuoto, così:

```
<WebService(Namespace:="")> _
        Public Class MyWebService
                …
        End Class
```

questo non sarà proprio conforme a tutti gli standards per i Web Services, ma ci consente di manipolare meglio il risultato con XPATH, senza questo accorgimento infatti viene inserito, nell'XML di output, un default namespace del tipo:

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns="http://mionamespace.org">
…
```

alcuni browser (come Firefox) hanno difficoltà nell'implementazione di query XPATH con un default namespace quindi meglio evitare…

LE FUNZIONI JAVASCRIPT

Di librerie AJAX da utilizzare con javascript ce n'è un po' per tutti i gusti. Una cosa che bisogna però tenere in considerazione è che il client (il browser dell'utente) quando si collega ad una pagina web non si scarica solo l'HTML e le immagini, ma anche i file javascript! Ciò significa che se una libreria, magari ottimamente costruita e ricca di funzioni, "pesa" 200kb gli utenti con una linea poco veloce percepiranno una generale lentezza del sito.
Inoltre c'è da dire che, per quanto complete e ben fatte, molte librerie AJAX richiedono uno studio ulteriore delle API, cioè delle funzioni da chiamare per utilizzarle, per cui, da una cosa relativamente semplice come il colloquio con HTTPRequest si finisce a dover studiare i Massimi Sistemi!
Dalle considerazioni di cui sopra, io sono arrivato alla considerazione di costruirmi una mia personale libreria (il file xml.js che trovate nei sorgenti) che:

1) "pesa" 11Kb
2) riproduce anche in Firefox i metodi e le funzioni della libreria XML di Microsoft, semplice da usare e ben documentata.

Oltre alla libreria di funzioni AJAX nei sorgenti si trovano anche altre librerie javascript che servono per non complicarsi troppo la vita nella stesura del codice. Ad esempio, la funzione usata per riferirsi a un elemento HTML a partire dal suo ID sarebbe:

```
document.getElementById("element1")
```

poiché questa è una funzione usata innumerevoli volte nel codice ho pensato a realizzare una libreria di supporto che contiene un Wrapper di questa funzione come:

```
function $(id) { return document.getElementById(id)}
```

In tal modo sarà sempre possibile utilizzare

```
$("element1")
```

invece della forma canonica, risparmiando sia in Kb dei file javascript che in battute di tastiera …
Questa ed altre analoghe scorciatoie o funzioni di supporto si trovano in un'altra libreria che utilizzo

AJAX: FU VERA GLORIA?

**AJAX, a mio modesto avviso, è un po' l' "invenzione dell'acqua calda" visto che, in realtà, non è altro che la manipolazione dei dati XML esterni con javascript.
Questo si poteva fare fin dalla versione 4 di Internet Explorer, solo che - nei siti web pubblici - abitualmente non si faceva perchè Internet Explorer era l'unico browser che forniva – seppur in forma proprietaria - strumenti XML da usare con Javascript.
Adesso si fa perchè anche altri browser (come Firefox) supportano XML e quindi le applicazioni sono sufficientemente cross-browser. Poi qualcuno ha avuto la bella idea di inventarsi il simpatico nome AJAX (acronimo per "Asynchronous JavaScript And XML") ed è nata l'ennesima moda...**

spesso per il DHTML chiamata core.js (anch'essa ovviamente presente nei sorgenti).
Naturalmente, per motivi di spazio, non possiamo metterci ad analizzare nel dettaglio tutte le funzioni presenti nelle librerie che includeremo nel nostro file HTML, ne parlo solo perché se analizzerete il sorgente, e negli esempi successivi, può esserci la chiamata a funzioni presenti in tali file.

L'IMPLEMENTAZIONE CONCRETA

Armati delle librerie "generiche", utili a svolgere le funzioni di interscambio e formattazioni dei dati, dobbiamo quindi implementare la logica specifica della nostra applicazione. Passiamo ad analizzare subito un esempio per comprendere meglio la logica di funzionamento del colloquio client/server.
Per prima cosa dobbiamo precisare che tutto, Web Services, pagine web, file javascript devono essere interni allo stesso sito web (questo perché, per motivi di sicurezza, non è possibile effettuare chiamate HTTPRequest in altri siti), quindi andiamo ad operare nella stessa directory dove, la volta scorsa, abbiamo messo i file dei Web Services e configurata come directory virtuale in IIS. Qui creiamo una cartella per gli scripts ed una per i CSS e, nella directory base inseriamo due nuovi file HTML:

I SORGENTI NEL CD

Nel CD allegato alla rivista si trova il codice sorgente dell'applicazione completo, sia per la parte dei Web Services che per il client AJAX illustrato nell'articolo.
Per quanto funzionante l'applicazione è da considerarsi più come un prototipo che come un programma; prima di poterla usare in produzione deve essere ancora testata a fondo e, nel client, devono essere implementate funzionalità (caricamento asincrono anziché sincrono e paginazione e filtri sulle tabelle tra tutte) indispensabili nel mondo reale.

- login.html – che presenta la maschera di login per l'utente non autenticato
- admin.html – che contiene il resto dell'applicazione

Prima di vedere come abbiamo risolto il problema dell'autenticazione dal lato client, vediamo come, attraverso javascript, possiamo richiamare dati dal Web Service.

Abbiamo implementato la funzione di login nel Web Service users.asmx in questo modo:

```
<WebMethod()> _
    Public Function Login(ByVal mail As String, ByVal
                        password As String) As UserResult
        Return DoOperation(Of User, UserResult)("",
         Mailer, "LoginUser", "users.login", mail, password)
    End Function
```

ricordiamo che **UserResult** non è altro che una delle diverse derivazioni di OperationResult ovvero il wrapper dell'oggetto effettivo restituito in forma serializzata e corredato da informazioni su eventuali errori.
In questo caso la funzione DoOperation richiama attraverso la reflection il metodo LoginUser dell'oggetto *Mailer* .
Questi sono tuttavia dettagli implementativi, andiamo subito a vedere il funzionamento del metodo Login aprendo il browser all'indirizzo *http://localhost/Mailing/users.asmx* (presupponendo naturalmente di avere configurato in IIS la directory virtuale chiamata Mailing in corrispondenza della path della nostra applicazione).
Dovrebbe quindi apparire la maschera di test di cui in **figura 1**.

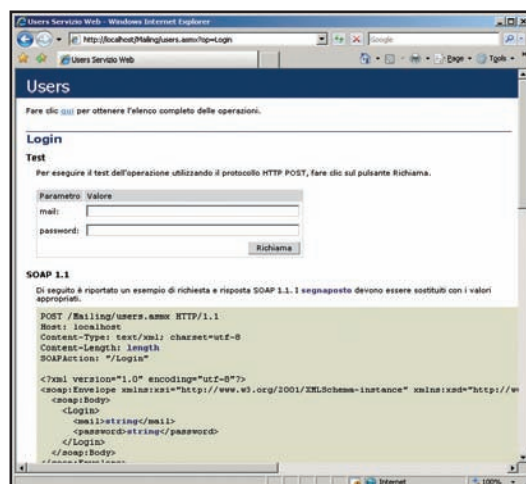



Fig. 1: ***Test del Web Service***

immettiamo mail e password e vediamo la risposta XML, che sarà:

```
<?xml version="1.0" encoding="utf-8" ?>
<UserResult
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                                            instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                                    success="true" />
```

ovvero con il solo elemento *UserResult*, nel caso in cui il login sia fallito (l'attributo success sta infatti ad indicare che l'operazione non ha causato eccezioni).
Mentre in caso di login positivo la risposta sarà:

```
<?xml version="1.0" encoding="utf-8" ?>
<UserResult
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-
```

```
instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
success="true">
<data>
  <user id="1" name="admin"
        mail="francesco@smelzo.it" power="100"
        password="fs" categories="2;1" />
</data>
</UserResult>
```

e quindi con un corpo costituito dall'elemento data al cui interno è serializzato l'oggetto della risposta (in questo caso user).
Quindi non resta che vedere come chiamare lo stesso metodo da javascript.
Il codice, che fa uso di varie funzioni presenti nelle librerie di supporto, è il seguente:

```
var users=new Object();
users.login = function (mail,password){
        return  loadSync("users.asmx /Login",
                        "mail=" +
                        mail.toString().toURI() +
                        "&password=" +
                        password.toString().toURI())
}
```

È appena il caso di dire che *loadSync* è una funzione definita nelle librerie di supporto che si occupa della chiamata HTTPRequest in forma sincrona.
I parametri mail e password vengono passati al Web Service attraverso la codifica URI standard (secondo parametro della funzione loadSync).
Il risultato della funzione sarà un XMLDocument navigabile con XPATH.
Questo codice viene utilizzato per implementare il sistema di autenticazione.

IL SISTEMA DI AUTENTICAZIONE

Viene quindi utilizzata una routine di autenticazione che controlla che nei Cookies sia presente il valore UID :

```
function checkAuth(){
        if(Cookies.get("UID",0)==0) {
                top.location = "login.html?ref="
                        + encodeURIComponent(top.location);
                return false
        }
        return true;
}
```

Se non trova il Cookie l'utente non è autenticato e quindi lo ridireziona verso la pagina di login aggiungendo all'url il parametro *ref* che rappresenta l'indirizzo della pagina che ha originato la richiesta.
Quindi, nella pagina che vogliamo proteggere (nel nostro caso admin.html) andremo a richiamare questa funzione sotto l'evento *onload*:

```
window.onload = function () {
        if ( checkAuth() ) {
                // codice di inizializzazione della pagina
        }
}
```

Se checkAuth fallisce la pagina di ridireziona appunto verso login.html. Quest'ultima presenta una maschera di login espressa in HTML in questo modo:

```
<form method="post" onsubmit="return
                formSubmit()">
        <table cellpadding="0" cellspacing="0"
                        class="panel">
                <tr align="left">
                        <td class="captionBar">
                                Login
                </tr>
                <tr align="left">
                        <td>
                        <div>E-mail</div>
                        <div><input
                type="text" id="mail" name="mail"/></div>
                        </td>
                </tr>
                <tr align="left">
                        <td>
                        <div>Password</div>
                        <div>
<input type="password" id="password"
name="password"/></div>
                        </td>
                </tr>
                <tr align="right">
```

PERCHÉ NON JSON?

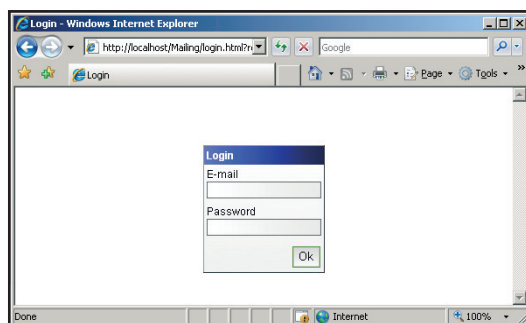
JSON (se ne è parlato in altri articoli di IoProgrammo) è un'altra tecnica di programmazione asincrona con javascript alternativa ad AJAX, mentre in AJAX i dati vengono forniti dal server attraverso XML in JSON (JavaScript Object Notation) vengono forniti direttamente come oggetti javascript serializzati.
Dalla sua JSON ha la maggiore leggerezza del flusso e, ovviamente, una maggiore integrazione con Javascript.
Di contro però la nostra applicazione Server dovrebbe "parlare" in JSON e non XML e quindi ci taglieremo fuori la possibilità di sviluppare client di altro tipo (dalla web application tradizionale fino alle applicazioni desktop, che invece possono interagire comodamente con i Web Services) .
E poi, diciamolo chiaramente, la gestione di XML non è poi così difficile, tanto più che, una volta imparata, ci può tornare utile in centinaia di ambiti diversi.

```
                    <td>
                    <div>
              <input type="submit" value="Ok"/></div></td>
                  </tr>
                  <tr align="left"><td id="result">
                  </td>
                  </tr>
       </table>
</form>
```

L'autenticazione avviene quindi all'interno della funzione formSubmit() che è espressa così:

```
function formSubmit(){
//chiamata del webservice attraverso users.login
                  var result =
      users.login($E.value("mail"),$E.value("password"));
//result è un XMLDocument
//viene individuato il nodo <user> con una query
                                                XPATH
                  var node = result.
                      selectSingleNode("//data/user");
                  if(!isNull(node)) {
//se il nodo viene trovato viene inserito il valore
                           // dell'attributo id nei cookies
                  var id = node.getAttribute("id");
                           Cookies.set("UID",id);
//redireziona verso la pagina originaria
                           window.location = ref;
                  } else {
//se il nodo non viene trovato l'elemento non esiste
//scrive in nell'elemento "result" il messaggio di  errore
$('result').innerHTML="login fallito!"
                  }
       return false
}
```

Fig. 2: *Maschera di login*

Francesco Smelzo è specializzato nello sviluppo in ambiente Windows con particolare riferimento ad applicazioni in ambiente .NET sia web-oriented che desktop. Il suo sito web è www.smelzo.it. Come sempre è a disposizione per ricevere suggerimenti o richieste sull'articolo all'indirizzo di posta elettronica francesco@smelzo.it

L'APPLICAZIONE VERA E PROPRIA

Questo per quanto attiene all'autenticazione, ma per tutto il resto? Dobbiamo ancora implementare un paio di funzionalità utili per la nostra applicazione. In particolare:

- utenti – aggiunta, modifica,cancellazione e assegnazione delle categorie di appartenenza
- ∑ categorie – aggiunta, modifica e cancellazione
- messaggi – aggiunta, cancellazione e visualizzazione del registro degli invii

Tutte queste attività vengono gestite unicamente all'interno del file admin.html, ognuna di esse ha la propria logica di funzionamento racchiusa in un file javascript esterno con un nome coerente (admin.html.users.js , admin.html.categories.js e admin.html.messages.js).
Le varie sezioni sono popolate on demand con il click sul tab corrispondente.
La sezione utenti presenta per prima cosa la tabella degli utenti che vediamo in **figura 3**.
Per mantenere compatta l'interfaccia ed evitare ricaricamenti di pagina, la gestione delle categorie viene attivata direttamente in una riga "a scomparsa" come vediamo in **figura 4**
La modifica dei dati e l'inserimento di nuovi utenti viene invece gestita con un finto popup (in realtà un DIV a scomparsa) come in **figura 5**.
La stessa interfaccia è stata naturalmente utilizzata anche nelle altre sezioni.
L'implementazione grafica vera e propria (costruzione dinamica delle tabelle, CSS ecc…) esula dagli scopi di questo articolo e non è importante ai fini della comprensione della tecnica, riportiamo qui solamente un esempio (semplificato) di rendering HTML, attraverso il DOM, di una lista di nodi XML:

```
function getCategoriesList(){
         //recupera XMLDocument dal Web Service
         var doc = categories.GetList();
         //seleziona con XPATH una lista di nodi
         var nodes =
                 doc.selectNodes("//data/list/category");
         //crea una tabella con il DOM
         var oTable =
                     document.createElement("table");
         //scorre i nodi XML della lista
         for(var i=0;i<nodes.length;i++){
                  var node = nodes[i];
                  //crea una riga nella tabella
                  var oTr = oTable.insertRow(-1);
                  //crea la prima cella della riga
```

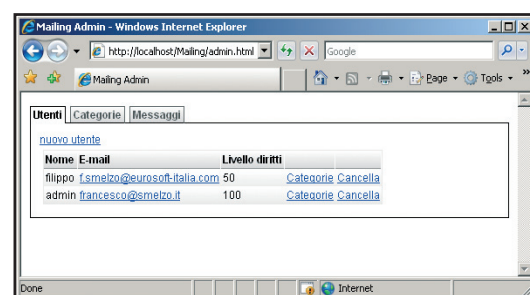



Fig. 3: *Sezione utenti*

```
                var oTd1 = oTr.insertCell(-1);
                //inserisce un valore preso da un
                        //attributo del nodo corrente
oTd1.innerHTML=node.getAttribute("name");
                //crea una seconda cella nella riga
                var oTd2 = oTr.insertCell(-1);
                oTd2.innerHTML= "Cancella";
        }
        //infine appende la tabella all'elemento DIV
                                          esistente
document.getElementById("catsList").appendChild(oT
                                              able);
}
```

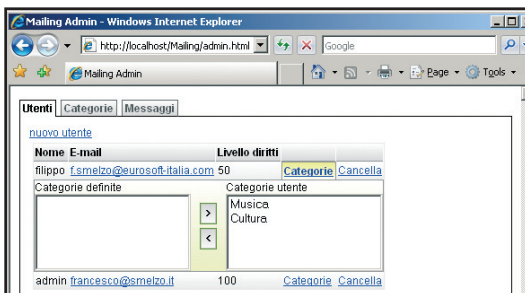



Fig. 4: ***Interfaccia assegnazione categorie a utenti***

Tutti i metodi esposti dai Web Services sono stati rimappati in corrispondenti funzioni javascript (come abbiamo visto per il login) – tecnicamente questo sistema si chiama proxing ed è simile a quello che avviene automaticamente quando si istanzia un web service in un progetto di Visual Studio.
Solo come esempio riportiamo un estratto di queste funzioni di mapping effettuato sul Web Service user.asmx:

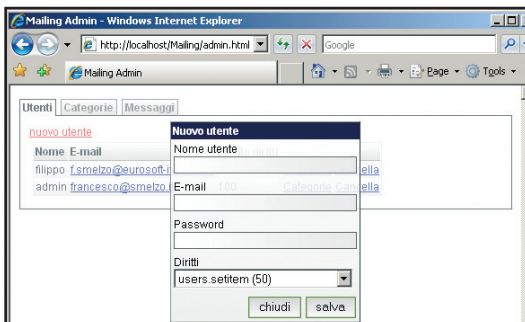
```
var users=new Object();
users.url = "users.asmx"
users.login = function (mail,password){
        return  loadSync(this.url + "/Login",
                        "mail=" +
                          mail.toString().toURI() +
                        "&password=" +
                      password.toString().toURI())
}
users.operatorId = function (){
```

```
        return Cookies.get("UID",0);}
users.GetList = function (query, order, action) {
        return loadSync (this.url + "/GetList",
                "operatorId=" + this.operatorId() +
              "&query=" + query.toString().toURI() +
              "&order=" + order.toString().toURI());
}
users.GetUserCategories = function (userID) {
        return
          loadSync(users.url + "/GetUserCategories",
        "operatorId=" + this.operatorId() +
        "&userID=" + userID.toString().toURI());
}
users.AddUserCategory = function
                        (userID,CategoryID) {
                return
          loadSync(users.url + "/AddUserCategory",
        "operatorId=" + this.operatorId() +
        "&userID=" + userID.toString().toURI() +
      "&CategoryID=" + CategoryID.toString().toURI());
}
users.RemoveUserCategory = function
        (userID,CategoryID) {
        return
        loadSync(users.url + "/RemoveUserCategory",
        "operatorId=" + this.operatorId() +
                "&userID=" +
         userID.toString().toURI() + "&CategoryID=" +
                CategoryID.toString().toURI());
}
```

Fig. 5: ***Editing utenti***

UN PICCOLO TRUCCO

L'operazione di mapping in javascript dei metodi esposti dal Web Service è molto utile per la stesura del codice ma è altrettanto noiosa.
Tuttavia ogni Web Service espone una descrizione WSDL della propria struttura raggiungibile con l'url http://mioserver/webservice.asmx? WSDL , tale descrizione è in formato XML quindi è possibile trasformare questa sorgente utilizzando XSLT in qualsiasi output, e quindi anche in codice javascript che non è altro che testo.

CONCLUSIONI

Abbiamo visto quindi come creare un client ricco di funzionalità con una semplice pagina HTML e un bel po' di codice javascript. Avendo dato questa struttura alla parte server del programma (Web Services) nulla ci impedisce di sviluppare altri tipi di client, da applicazioni Web più "tradizionali" fino a client desktop veri e propri mantenendo la stessa logica applicativa.

Francesco Smelzo

Apache mod_rewrite
http://httpd.apache.org/docs/1.3/mod/mod_rewrite.html
Sulle regular expressions
http://en.wikipedia.org/wiki/Regular_expression
http://regexlib.com/

SVILUPPARE UNA CHAT CON AJAX E PHP

UNO DEGLI UTILIZZI PIÙ DIVERTENTI DI AJAX È SICURAMENTE LA CREAZIONE DI UNA CHAT. ECCO COME APPROFITTARE DEL JAVASCRIPT ASINCRONO PER CREARE CHAT PROFESSIONALI SUL WEB CON PHP SENZA SCOMODARE PER FORZA JAVA O MACROMEDIA FLASH !

La grande innovazione apportata da Ajax è, principalmente, legata al fatto che tale tecnologia consente di adottare anche nell'ambito dello sviluppo web un approccio molto simile a quello tipico della programmazione *desktop*.
Le prime applicazioni *web-oriented*, in effetti, erano costituite da singoli programmi che venivano eseguiti all'occorrenza su richiesta dell'utente: il server si limitava a processare la richiesta generando l'output da consegnare al client, ma al termine di tale procedura basata su domanda-risposta la memoria del server veniva svuotata e non rimaneva più alcuna traccia della transazione. E' proprio per questo motivo che si dice che il protocollo HTTP è privo di stato, o "*sessionless*" (per dirlo all'inglese).
La realizzazione di una vera e propria *web-application* (con tutte le caratteristiche che essa comporta) era infatti appannaggio della sola nascente piattaforma Java, penalizzata tuttavia da molti errori di gioventù (lentezza, difficoltà di utilizzo, scarsa stabilità), mentre la riproduzione degli stessi meccanismi (o quantomeno di meccanismi simili) con i linguaggi di scripting tradizionali (Asp, Php), era affidata più che altro alla fantasia e all'estro del singolo programmatore.
E' vero che presto anche i linguaggi di scripting si sono evoluti, integrando funzionalità avanzate per tenere traccia, ad esempio, delle sessioni lato-server aperte per tutta la durata della navigazione da parte dell'utente, tuttavìa la necessità di appoggiarsi al protocollo HTTP per consentire l'interazione tra client e server, di fatto, continuava a rappresentare un ostacolo insormontabile alla realizzazione di applicazioni veramente efficienti.
Ed è proprio sull'onda di queste necessità che si è fatta avanti la tecnologia Ajax, in grado di superare i limiti della programmazione *web-oriented* tradizionale per permettere invece un approccio allo sviluppo su Internet molto più facile e performante.
Immaginiamo, ad esempio, di dover realizzare una chat, sul modello della Gmail Chat di Google o simili: se dovessimo farlo con un linguaggio di scripting tradizionale, come Php, sicuramente incontreremmo qualche ostacolo che ci costringerebbe ad adottare degli stratagemmi particolari.

HTTPXMLREQUEST: ISTRUZIONI PER L'USO

E' evidente che lo sviluppo della chat in modalità "classica" obbliga il programmatore a usare dei "trucchetti" particolari, come inserire un tag META nell'intestazione del documento html, che effettua un "refresh" automatico della pagina ogni tot secondi (un'altra possibilità sarebbe anche inserire un timer con javascript che effettua il reload della pagina, ma personalmente tra le due alternative preferisco la prima). Il risultato è, tuttavìa, alquanto deludente se non si dispone di una connessione ad Internet sufficientemente veloce, senza contare che per poter aggiornare solo una parte della pagina è necessario lavorare con i frame, con tutti i problemi che questi comportano, ovviamente.
Per ovviare a tali inconvenienti, ecco quindi la necessità di lavorare con Ajax, il quale ci consente di modificare dinamicamente certe parti della pagina senza dover effettuare il *reload* della stessa. Un sito sviluppato con Ajax è molto simile ad un'applicazione installata fisicamente su un client che effettua chiamate ad un server: la velocità della connessione è molto superiore rispetto a quella dei siti classici, mentre l'occupazione della banda si riduce notevolmente incrementando, di fatto, la resa e la produttività dell'applicazione stessa.
Per capire come funziona un'applicazione

Ajax occorre conoscere il funzionamento dell'oggetto XmlHttpRequest, che in questa sede darò in parte per scontato al fine di concentrare l'attenzione solo sugli aspetti più significativi che contraddistinguono questa tecnologia. In realtà esistono ormai diverse librerie che affrontano l'argomento anche con approccio decisamente avanzato, fornendo talvolta framework completi in grado di semplificare al massimo lo sviluppo di applicazioni complesse. Tuttavìa, soprattutto per lavori semplici come quello che stiamo per sviluppare, affidarsi ad una specifica libreria può essere alquanto vincolante per il programmatore, mentre lavorare direttamente con XmlHttpRequest offre maggiore libertà di movimento e alleggerisce notevolmente le dimensioni dell'applicazione.
Analizziamo ora le differenze tra un'applicazione standard e una che fa uso, invece, di Ajax: osserveremo che nel primo caso, su pressione di un pulsante Submit, l'intera pagina si bloccherà (per tutto il tempo occorrente a terminare la transazione con il server), quindi scomparirà e verrà completamente sostituita da quella effettivamente desiderata. Con Ajax, invece, la nostra richiesta remota avverrà in modo completamente asincrono, ossia slegato dalla coda degli eventi della pagina che continuerà pertanto a rispondere alle nostre sollecitazioni, anche nel caso di una transazione con il server particolarmente lunga. E' vero che si è instaurata una comunicazione con il server, ma ciò avviene in modo del tutto trasparente agli occhi dell'utente, che quindi può continuare a utilizzare in tutta libertà gli altri link, pulsanti, ecc…
Naturalmente la comunicazione asincrona con il server avviene tramite il sopracitato oggetto XmlHttpRequest, che com'è ovvio deve essere supportato dal browser che lo utilizza. A questo riguardo, va rimarcato che la maggior parte dei browser supporta XmlHttpRequest nativamente, tranne Internet Explorer fino alla versione 6.0 che invece utilizza al suo posto un ActiveXObject (non vi sono comunque differenze di rilievo nel comportamento dei due oggetti, cambia soltanto il codice Javascript che li implementa).
Un'altra considerazione importantissima di cui tenere conto è la necessità di sincronizzare i metodi che utilizzano XmlHttpRequest: è bene puntualizzare, infatti, che se l'oggetto è già impegnato in una transazione, non è possibile cercare di riutilizzarlo fino a che questa non è terminata: in caso contrario verrà generato un errore e la richiesta, ovviamente, fallirà. Al fine di creare applicazioni thread-safe, pertanto, è necessario interrogare lo stato del parametro readyState, che può assumere diversi valori:

- 0 **Uninitialized**: l'oggetto esiste, ma non è ancora stata posta in essere una comunicazione;
- 1 **Open**: è stato richiamato il metodo open(), per instaurare una comunicazione;
- 2 **Sent**: il metodo send() è stato eseguito ed ha inviato i dati (tramite GET o POST);
- 3 **Receiving**: la risposta è in fase di ricezione da parte del client;
- 4 **Loaded**: l'operazione termina: è l'unico stato, in pratica, che è supportato allo stesso modo da tutti i browser, e consente di sapere quando l'oggetto XmlHttpRequest è libero per poter essere riutilizzato.

Ma come facciamo in un'applicazione pratica, com'è la nostra chat, a gestire correttamente lo stato di XmlHttpRequest senza correre il rischio di perdere delle comunicazioni ?
Se tentiamo, infatti, di usare tale oggetto mentre è già occupato, e ci limitiamo a controllare preventivamente il suo stato, tutto ciò che otterremo in caso di esito negativo del controllo è evitare di incorrere in errori di Ajax, ma non avremo comunque risolto il problema della richiesta che non è andata a buon fine. Per risolvere questo problema abbiamo quindi diverse alternative:

- se non è poi così importante la perdita di tale richiesta, possiamo semplicemente limitarci a gestire l'errore;
- altrimenti possiamo creare più istanze dell'oggetto XmlHttpRequest, una per ogni richiesta, avendo quindi la certezza che queste non andranno in conflitto tra di loro;
- visto che creare troppe istanze dell'oggetto XmlHttpRequest può rallentare molto l'esecuzione del codice, allora forse è meglio, in caso di oggetto già impegnato, eseguire un'istruzione setTimeout che, dopo un certo

REQUISITI PER L'UTILIZZO DEI FILE DI ESEMPIO

Sul CD allegato è disponibile una cartella che contiene la chat oggetto dell'articolo. Per poter far funzionare correttamente l'applicazione occorre aver installato sul proprio PC Php versione > 5.0, MySql e Apache. E' richiesto anche il supporto alla libreria Mysqli (mysql improved), che fornisce un accesso a MySql orientato agli oggetti. Per abilitare la libreria (sempreché ovviamente essa sia presente tra le estensioni attivabili), è sufficiente di solito decommentare (o eventualmente aggiungere) la riga "extension=php_mysqli.dll" nel php.ini. Una volta copiata la cartella nella directory di Apache, quindi, bisogna creare un nuovo db di nome Ajax ed eseguire lo script Sql per creare la tabella.

tempo prefissato, riesegue la chiamata alla stessa funzione in modo asincrono per verificare se l'oggetto nel frattempo si è disimpegnato;

- il metodo appena illustrato è perfetto nella maggior parte dei casi perché consente ad una funzione che cerca di impegnare un oggetto XmlHttpRequest di richiamare se stessa per un numero indefinito di volte fino a che questa non trova l'oggetto libero; ciò che tuttavia esso non è in grado di controllare è l'esatto ordine delle richieste inviate, che si può avere solamente implementando un meccanismo simile ad una coda. In pratica, prima di inviare le richieste, queste vengono impilate in un array di Javascript; la funzione che invia le richieste al server, quindi, comincerà la propria attività da quella più vecchia (sfruttando il principio del FIFO -> First In, First Out), quindi al termine del processamento di questa passerà alla seconda, ecc… In questo modo siamo certi che verrà rispettato l'esatto ordine delle richieste.

Nel caso della nostra chat, in particolare, ci avvarremo di un oggetto XmlHttpRequest, che gestisce l'invio e la ricezione dei messaggi. Per sincronizzare l'accesso all'XmlHttpRequest sfrutteremo una combinazione degli ultimi due metodi appena illustrati (setTimeout + cache): su pressione del pulsante di Invio del messaggio, il contenuto della casella di testo non sarà direttamente inviato al server, ma verrà accodato all'interno di un array; periodicamente, poi, verrà eseguita una funzione che preleva di volta in volta l'elemento della coda più vecchio e lo spedisce al server (sempre ché l'oggetto XmlHttpRequest non sìa già impegnato altrove, altrimenti l'istruzione setTimeout si limita a ritentare la chiamata alla stessa funzione in modo asincrono dopo un certo periodo, per verificare nuovamente lo stato dell'oggetto).
Non ci resta ora che gestire il risultato ritornato dall'elaborazione lato-server: è possibile, a questo scopo, utilizzare il parametro responseText, che recupera la risposta senza tenere conto della reale natura della stessa (file di testo, XML, html, ecc…).
In questo modo, semplicemente, possiamo usare il metodo innerHtml di un div per aggiornare dinamicamente il suo contenuto sulla base del testo generato dal server. Ma possiamo anche richiedere a quest'ultimo di generare un file XML come risposta da ritornare al client: il formato XML, come sappiamo bene, è estremamente versatile, e ci consente il rapido reperimento delle informazioni che ci servono semplicemente effettuando un'operazione di parsing (sempreché tale file sia formattato correttamente). Nel caso della nostra chat mi sono avvalso di questa seconda opportunità, utilizzando quindi il parametro responseXml. Non resta ora che buttare giù qualche riga di codice, tuttavia per ragioni di spazio mi limito a riportare solo il codice Javascript più importante e la funzione PHP che gestisce le varie richieste effettuate dall'utente:

```
// questa funzione esegue chiamate asincrone nei
                                   confronti del server
// allo scopo di ottenere i  nuovi messaggi
function aggiorna_chat()
{
  var utente =
        document.getElementById("utente").value;
  var parametri = "";
    // se l'oggetto XmlHttpRequest è stato istanziato
                                    correttamente...
  if(xml_Messaggi)
  {
    try
    {
          // il presente controllo evita che si cerchi
               di effettuare una nuova operazione
          // con l'oggetto xmlHttp_Messaggi se
                           questo è già impegnato
      if (xml_Messaggi.readyState == 4 ||
          xml_Messaggi.readyState == 0) // il
             controllo viene in ogni caso eseguito
                  // se l'oggetto esiste ma non è
                                 stato ancora utilizzato
      {
          // estraggo dalla cache il messaggio più
             vecchio, se questa contiene messaggi
                  // La cache funziona un po' come
  la coda che facciamo al supermercato: il primo che
                  // arriva ovviamente è anche il
       primo ad essere servito, cosicché siamo sicuri
                  // che i messaggi vengono
    ricevuti nell'ordine esatto in cui essi sono inviati
                  // Se la cache è piena estraggo
    l'elemento più vecchio allocato in essa e mando
                  // la richiesta al server di
visualizzarlo nella chat, recuperandolo tramite l'id del
                  // messaggio. In caso contrario,
   semplicemente richiedo al server i nuovi messaggi
```

COS'È AJAX ?

Ajax è l'acronimo di Asynchronous JavaScript And XML (Javascript asincrono e XML). Tale denominazione, usata per la prima volta da Jesse Garrett nel 2005 all'interno di un suo blog, sta a sottolineare come l'obiettivo di questa nuova concezione preveda l'utilizzo di Javascript asincrono interfacciato con XML, migliorando quindi notevolmente l'interazione tra client e server e avvicinando sempre di più il mondo dello sviluppo web classico a quello molto più specializzato delle applet Java o delle tecnologie Macromedia Flash.

```
        if (cache.length>0)
        {
          parametri = cache.shift();
        }
        else
        {
          parametri = "richiesta=nuovi_messaggi&id="
                                              +ultimo_id;
        }
        xml_Messaggi.open("POST", "chat.php", true);
        xml_Messaggi.setRequestHeader("Content-
         Type", "application/x-www-form-urlencoded");
        xml_Messaggi.onreadystatechange =
                                         gestisci_risposta;
         xml_Messaggi.send(parametri);
      }
      else
      {
        setTimeout("aggiorna_chat();", intervallo);
      }
    }
    catch(e)
    {
      alert("Errore nel tentativo di recuperare i nuovi
                                          messaggi ...");
    }
  }
}
/* questa funzione PHP è specializzata nel gestire le
                          richieste dei nuovi messaggi */
  public function restituisci_nuovi_messaggi($id)
  {
    $id = $this->connessione-
                               >real_escape_string($id);
    if($id>0)
    {
      $query =
       'select id_messaggio, utente, messaggio,
 date_format(ora, "%Y-%m-%d %H:%i:%s") as ora '
                                                      .
       ' from chat where id_messaggio > ' . $id .
       ' order by id_messaggio asc';
    }
    else
    {
      $query =
       ' select * from chat order by id_messaggio asc';
    }
    $result = $this->connessione->query($query);
    $risposta = '<?xml version="1.0"
                 encoding="UTF-8" standalone="yes"?>';
    $risposta .= '<risposta>';
    if($result->num_rows)
    {
      while ($riga = $result-
                             >fetch_array(MYSQLI_ASSOC))
      {
        $id = $riga['id_messaggio'];
        $utente = htmlspecialchars ($riga['utente']);
        $ora = htmlspecialchars ($riga['ora']);
        $messaggio = htmlspecialchars
                                  ($riga['messaggio']);
        $risposta .= '<id>' . $id . '</id>' .
                     '<ora>' . $ora . '</ora>' .
                     '<utente><![CDATA[' . $utente .
                                        ']]></utente>' .
                     '<messaggio><![CDATA[' .
   $messaggio . ']]></messaggio>'; //con CDATA non
                                   dobbiamo preoccuparci
                                   //dei simboli che
                                   inseriamo nell'XML,
                                   //siano essi entità o
                                                   altro
      }
      $result->close();
    }

    $risposta = $risposta . '</risposta>';
    return $risposta;
  }
```

COSA SONO I FRAME?

I frame, che ho utilizzato nella prima chat di esempio di questo articolo, consentono di suddividere la pagina web in più riquadri, che possono puntare ciascuno ad una pagina web diversa. Essi consentono, ad esempio, di effettuare il refresh di uno solo di questi riquadri, senza dover per forza ricaricare tutta la pagina. Accanto a questo incontestabile pregio, tuttavia, l'uso dei frame presenta tutta una serie di difetti legati soprattutto alla complessità di progettazione della pagina, e al fatto che la suddivisione in più riquadri indipendenti gli uni dagli altri rompe la struttura classica del documento Html, generando vari problemi di interpretazione da parte di alcuni browser in certe situazioni (es. il salvataggio della pagina nei segnalibri, ecc...). Nel 1996 Microsoft introdusse in Internet Explorer 3.0 gli iframe, più semplici e immediati nell'utilizzo rispetto ai precedenti, che subito alcuni programmatori impiegarono per simulare (in maniera abbastanza sporca, tuttavia) un'interazione trasparente con il server: ciò avveniva semplicemente modificando l'attributo src della pagina inclusa nell'iframe in questione.

CONCLUSIONI

Gli esempi presenti nel CD, lasciati volutamente abbastanza scarni dal punto di vista delle funzionalità trattate, abbracciano tuttavìa la maggior parte delle problematiche connesse con la programmazione Ajax, e consentono, con facilità, di iniziare subito a lavorare con questa intrigante filosofia legata allo sviluppo web: una filosofia che, sicuramente, costituirà il futuro dell'intero mondo delle Rich Internet Applications.

Enrico Viale

IIS COME APACHE CON L'URL REWRITING

SI TRATTA DI UNA DELLE FUNZIONALITÀ PIÙ AMATE DAGLI UTILIZZATORI DEL NOTO WEB SERVER OPENSOURCE. CONSENTE DI ELIMINARE I PARAMETRI LUNGHISSIMI CHE SOLITAMENTE CARATTERIZZANO UNA QUERY_STRING. CREIAMONE UNO PER IIS

Ormai la maggior parte dei siti web si appoggia a database per quanto riguarda i contenuti. Se, da un lato, si hanno notevoli vantaggi in termini di funzionalità e semplicità di gestione, è anche vero che c'è un prezzo da pagare... Pensiamo al sito web di una società che vende prodotti hi-tech online. Tipicamente i prodotti saranno suddivisi in categorie e sottocategorie (es. software – antivirus). La pagina che visualizza i prodotti di una sottocategoria sarà ad esempio: http://nomesito.com/prodotti.aspx?cat_id=104&sca_id=827. Url di questo tipo presentano due inconvenienti:

- sono difficilmente memorizzabili dall'utente
- sono penalizzanti nell'indicizzazione sui motori di ricerca

Dal punto di vista dell'utente, sarebbe meglio poter digitare un url del tipo http://nomesito. com/prodotti/software/antivirus/ invece di quello visto sopra. Inoltre, probabilmente la società proprietaria del sito spenderà consistenti cifre in pubblicità e destinerà una porzione di budget all'indicizzazione sui motori di ricerca. Molti spider faticano ad indicizzare correttamente pagine dinamiche, preferendo url statici che non fanno uso di caratteri come ?, &, +, = ed altri legati alla costruzione classica di pagine dinamiche. Una possibile soluzione al problema consiste nell'implementare un modulo http che "intercetti" le richieste fatte al server, le "decodifichi" in base a delle regole e "rediriga" l'utente verso la risorsa desiderata. Questa tecnica prende il nome di "URL Rewriting".

REQUISITI

Conoscenze richieste
SQL Server, .Net Framework 2.0

Software
SQL Server 2005, Visual Studio 2005

Impegno

Tempo di realizzazione

CONVERTIAMO GLI INDIRIZZI

Lo scopo dell'articolo è abbastanza semplice. L'applicazione riceverà URL del tipo http://nomesito/param1/param2 dove param1 e param2 sono percorsi fisicamente inesistenti, e li convertirà in indirizzi comprensibili e reali del tipo http://nomesito?param1=param2

Prima di partire con lo sviluppo del modulo realizziamo lo scheletro dell'applicazione su cui quest'ultimo verrà testato. Apriamo Visual Studio e creiamo un nuovo sito web in linguaggio Visual Basic e chiamiamolo RewriteTestVB. Aggiungiamo al progetto un file di configurazione per l'applicazione (web.config) e nella sezione <**configuration**> scriviamo:

```
<configSections>
<sectionGroup name="mod_rewrite">
    <section name="simple_rules" type="System.
        Configuration.NameValueSectionHandler" />
<section name="regex_rules" type="System.
        Configuration.NameValueSectionHandler" />
    </sectionGroup>
</configSections>
```

Abbiamo definito delle nuove sezioni del web.config (simple_rules e regex_rules) che ci serviranno per inserire le regole. Inseriamo quindi i nostri set di regole tramite delle coppie chiave valore, in cui

- key = url digitato
- value = url vero su cui fare il rewrite

```
<mod_rewrite>
    <simple_rules>
        <add key="/RewriteTestVB/Home/"
    value="/RewriteTestVB/Default.aspx?tab=Home" />
        <add key="/RewriteTestVB/Home"
    value="/RewriteTestVB/Default.aspx?tab=Home" />
        <add key="/RewriteTestVB/ChiSiamo/"
  value="/RewriteTestVB/Default.aspx?tab=ChiSiamo "/>
        <add key="/RewriteTestVB/ChiSiamo"
  value="/RewriteTestVB/Default.aspx?tab=ChiSiamo" />
        <add key="/RewriteTestVB/DoveSiamo/"
  value="/RewriteTestVB/Default.aspx?tab=DoveSiamo" />
        <add key="/RewriteTestVB/DoveSiamo"
  value="/RewriteTestVB/Default.aspx?tab=DoveSiamo" />
        <add key="/RewriteTestVB/Contattaci/"
  value="/RewriteTestVB/Default.aspx?tab=Contattaci" />
```

```
            <add  key="/RewriteTestVB/Contattaci"
value="/RewriteTestVB/Default.aspx?tab=Contattaci" />
            <add  key="/RewriteTestVB/Categorie/"
                value="/RewriteTestVB/Categorie.aspx" />
            <add key="/RewriteTestVB/Categorie"
                value="/RewriteTestVB/Categorie.aspx" />
        </simple_rules>
        <regex_rules>
            <add key="/RewriteTestVB/Sottocategorie/(.*)/"
 value="/RewriteTestVB/Sottocategorie.aspx?cat=$1" />
            <add key="/RewriteTestVB/Sottocategorie/(.*)"
 value="/RewriteTestVB/Sottocategorie.aspx?cat=$1" />
            <add key="/RewriteTestVB/Prodotti/(.*)/(.*)/"
        value="/RewriteTestVB/Prodotti.aspx?cat=$1&amp
                                        ;scat=$2" />
            <add key="/RewriteTestVB/Prodotti/(.*)/(.*)"
        value="/RewriteTestVB/Prodotti.aspx?cat=$1&amp
                                        ;scat=$2" />
        <add  key="/RewriteTestVB/DettagliProdotto/(.*)/"
value="/RewriteTestVB/DettagliProdotto.aspx?pro=$1" />
        <add key="/RewriteTestVB/DettagliProdotto/(.*)"
value="/RewriteTestVB/DettagliProdotto.aspx?pro=$1" />
        </regex_rules>
</mod_rewrite>
```

Abbiamo definito due set di regole. Il primo, chiamato simple_rules, contiene le regole statiche che mappano le pagine Home, ChiSiamo, DoveSiamo, Contattaci e Categorie con la Default.aspx a cui viene passato un valore “tab” nella QueryString.
Il secondo, regex_rules, contiene le regole dinamiche, create utilizzando delle semplici regular expressions. Prendiamo, ad esempio, la regola che mostra i prodotti:

- Prodotti/(.*)/(.*)/
- Prodotti.aspx?cat=$1&scat=$2

Permette di trasformare url del tipo Prodotti/CODICE1/CODICE2/ in url del tipo Prodotti.aspx?cat=CODICE1&scat= CODICE2. Per il momento possiamo chiudere il file web.config. Adesso che lo scheletro dell’applicazione è pronto, passiamo alla realizzazione del modulo HTTP che si occuperà di effettuare il “rewrite” degli url.

HTTPHANDLERS ED HTTPMODULES

I Web Server utilizzano dei componenti per estendere le proprie funzionalità base. IIS, ad esempio, utilizza una tecnologia chiamata ISAPI (acronimo di Internet Server API) ed, in particolare, le “ISAPI Extensions” e gli “ISAPI Filters”. Le prime sono delle applicazioni che estendono le funzionalità del server: vengono implementate tramite delle dll e possono essere eseguite in seguito a richieste del client; i secondi sono dei filtri applicati alle richieste http. Microsoft .Net fornisce HttpModules ed HttpHandlers per realizzare questo tipo di funzionalità. Quando una richiesta arriva al Web Server, viene processata da eventuali moduli HTTP, gestita da un solo handler ed, infine, la risposta viene inviata al client.

CREIAMO IL NOSTRO MODULO HTTP

I moduli HTTP sono dei componenti che implementano l’interfaccia IHttpModule del namespace System.Web. Questa interfaccia contiene due metodi che sono: Init e Dispose. Il metodo Init consente al modulo di registrare i gestori di eventi dell’applicazione; il Dispose viene eseguito per liberare le risorse. Aggiungiamo un nuovo progetto di tipo “Visual Basic Class Library” alla solution creata. Per farlo, clicchiamo con il tasto destro del mouse sul nome della solution e selezioniamo “Add” -> “New Project”; dalla maschera successiva scegliamo il tipo di progetto e diamogli come nome “mod_rewrite_vb” Rinominiamo Class2.vb in Rewriter.vb. Ecco il codice della classe:

```
Imports System
Imports System.Web
Imports System.Collections.Specialized
Imports System.Configuration
Imports System.Text.RegularExpressions
Public Class rewriter
        Implements IHttpModule
        Public Sub Init(ByVal App As
            HttpApplication) Implements IHttpModule.Init
                AddHandler App.BeginRequest,
                            AddressOf Rewrite_Url
        End Sub
        Public Sub Rewrite_Url(ByVal sender As
                Object, ByVal args As System.EventArgs)
                Dim App As HttpApplication =
                        CType(sender, HttpApplication)
                Dim PageName As String =
                            App.Request.Path.ToLower()
                'Regole semplici
                Dim simple_rules As
                            NameValueCollection =
    CType(ConfigurationManager.GetSection("mod_rew
        rite/simple_rules"), NameValueCollection)
                If Not simple_rules Is Nothing Then
                    For i As Integer = 0
                            To simple_rules.Count - 1
                                Dim Source
        As String = simple_rules.GetKey(i).ToLower()
                                Dim
        Destination As String = simple_rules.Get(i)
                            If PageName = Source Then
```

```
                App.Context.RewritePath(Destination)
                Exit For
            End If
        Next
    End If
    'Regex
    Dim regex_rules As
        NameValueCollection =
    CType(ConfigurationManager.GetSection("mod_rew
        rite/regex_rules"), NameValueCollection)
    If Not regex_rules Is Nothing Then
        For i As Integer = 0
            To regex_rules.Count - 1
            Dim Rule As
        Regex = New Regex(regex_rules.GetKey(i),
            RegexOptions.IgnoreCase)
            Dim Match1
        As Match = Rule.Match(PageName)
            If  Match1.Success Then
App.Context.RewritePath(Rule.Replace(PageName,
            regex_rules.Get(i)))
Exit For
            End If
        Next
    End If
End Sub
Public Sub Dispose() Implements
            IHttpModule.Dispose
End Sub
End Class
```

Fig. 2: ***Creiamo il nostro HttpModule***

Abbiamo importato quattro namespace:

- **System.Web**: contiene l'interfaccia IHttpModule.
- **System.Configuration**: contiene la classe *ConfigurationManager* che ci permetterà di leggere dal web.config.
- **System.Collections.Specialized**: contiene la classe *NameValueCollection* per leggere le sezioni del web.config.
- **System.Text.RegularExpressions**: per le espressioni regolari.

La classe implementa l'interfaccia IHttpModule e deve quindi definire i metodi Init e Dispose.
Il metodo Init è molto semplice, contiene infatti una sola riga:

```
AddHandler App.BeginRequest, AddressOf   Rewrite_Url
```

che registra un gestore per l'evento BeginRequest, associandogli il metodo Rewrite_Url. Il metodo verrà attivato ad ogni richiesta di pagina.
Il metodo Rewrite_Url è il nucleo del modulo: si occupa di leggere e processare le regole, e, in caso di corrispondenze tra gli url digitati e le pagine richieste, riscrivere i path.
Nelle prime due righe del metodo:

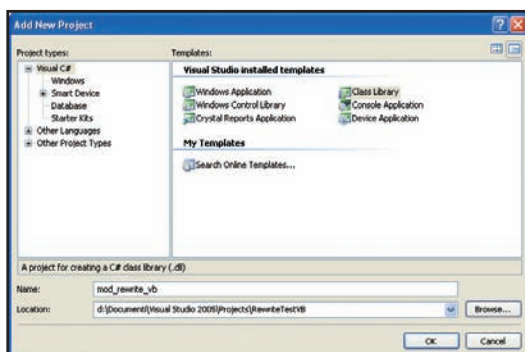
```
Dim App As HttpApplication = CType(sender,
                                HttpApplication)
Dim PageName As String =
                    App.Request.Path.ToLower()
```

instanziamo un oggetto di tipo HttpApplication e lo utilizziamo per ottenere il percorso della pagina richiesta al server.
Abbiamo poi due sezioni distinte: la prima si occupa di leggere le regole statiche; la seconda di leggere quelle che fanno uso delle espressioni regolari.
In entrambe instanziamo un oggetto di tipo **NameValueCollection** che conterrà i valori presenti nelle section create in precedenza nel web.config.
Ne "cicliamo" quindi il contenuto per controllare se ci sono corrispondenze.
Per le regole semplici:

```
Dim Source As String =
            simple_rules.GetKey(i).ToLower()
Dim Destination As String = simple_rules.Get(i)
If PageName = Source Then
        App.Context.RewritePath(Destination)
        Exit For
End If
```

controlliamo se la pagina che abbiamo digitato è presente tra le regole (leggendo il valore key della regola) e, se lo è, riscriviamo il path (utilizzando il path presente nel campo value della regola).
Per le regole dinamiche invece:

NOTA

Tutto il codice dell'articolo è scritto in Visual Basic, nel CD allegato alla rivista è comunque contenuto l'intero progetto scritto in C#.

APACHE E MOD_REWRITE

Apache HTTP Server è uno dei Web Server più diffusi al mondo. Nasce nel 1995 grazie al lavoro di una fondazione no-profit chiamata "Apache Software Foundation" e si diffonde in tutto il mondo per la sua affidabilità e le elevate performance. Tra i moduli più utilizzati in Apache ne troviamo uno chiamato mod_rewrite che utilizza un motore di riscrittura basato su un parser di regular expression.

```
Dim Rule As Regex = New
                    Regex(regex_rules.GetKey(i),
                        RegexOptions.IgnoreCase)
Dim Match1 As Match = Rule.Match(PageName)
If Match1.Success Then
App.Context.RewritePath(Rule.Replace(PageName,
                                regex_rules.Get(i)))
        Exit For
End If
```

utilizziamo la classe Regex per controllare se le regole scritte tramite regular expressions hanno un "match" con la pagina richiesta e, in caso affermativo, viene effettuato il rewrite.
In entrambi i casi utilizziamo il metodo RewritePath per riscrivere dinamicamente il path.

CONFIGURIAMO LA WEB APPLICATION

Una volta creato il modulo HTTP, occorre registrarlo nella nostra applicazione web. Farlo è semplice, basta aggiungere poche righe nel file web.config.
Compiliamo la libreria mod_rewrite_vb.dll ed aggiungiamone la reference al progetto RewriteTestVB. Per farlo, clicchiamo con il tasto destro del mouse sulla solution e quindi su "Add" -> "Reference". Clicchiamo su "Browse" e scegliamo il

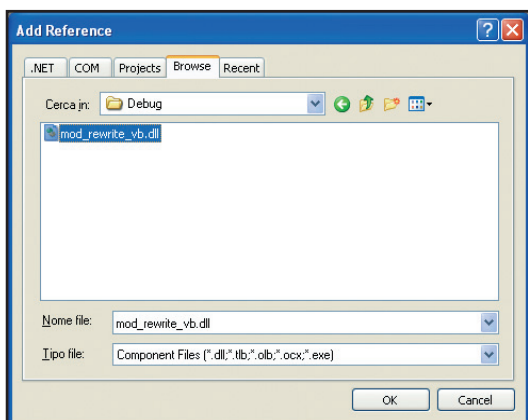



Fig. 3: ***Importiamo la libreria***

file mod_rewrite_vb.dll. Importata la libreria, occorre dire all'applicazione che deve utilizzarla come modulo HTTP. Per farlo, apriamo il file web.config ed aggiungiamo le seguenti righe nella sezione system.web:

```
<httpModules>
        <add
    type="mod_rewrite_vb.rewriter,mod_rewrite_vb"
                        name="mod_rewrite_vb" />
</httpModules>
```

l'attributo type specifica il tipo di modulo HTTP in termini di classe ed assembly; l'attributo name specifica il nome del modulo.
Una volta registrato l'assembly, possiamo creare le pagine del nostro sito dei prodotti.
Come prima cosa, creiamo un controllo "Menu.ascx" che inseriremo in tutte le pagine.

```
<%@ Control Language="VB"
    AutoEventWireup="false" CodeFile="Menu.ascx.vb"
                            Inherits="Menu" %>
<asp:Menu id="Menu1" runat="server"
        ForeColor="#284E98" Font-Names="Verdana"
    StaticSubMenuIndent="10px" Font-Size="X-Small"
                    DynamicHorizontalOffset="2"
                        BackColor="#B5C7DE">
<StaticMenuItemStyle Font-Names="Verdana" Font-
        Size="X-Small" HorizontalPadding="5px"
        VerticalPadding="2px"  />
<DynamicHoverStyle BackColor="#284E98"
                        ForeColor="White"  />
<DynamicMenuStyle BackColor="#B5C7DE"  />
<StaticSelectedStyle BackColor="#507CD1"  />
<DynamicSelectedStyle BackColor="#507CD1"  />
<DynamicMenuItemStyle HorizontalPadding="5px"
                        VerticalPadding="2px"  />
<Items>
        <asp:MenuItem
    NavigateUrl="/RewriteTestVB/Home" Text="Home
                    Page" Value="Home Page">
        </asp:MenuItem>
        <asp:MenuItem Text="Societ&#224;"
                        Value="Societ&#224;">
                <asp:MenuItem
    NavigateUrl="/RewriteTestVB/ChiSiamo" Text="Chi
                    Siamo" Value="Chi Siamo">
                </asp:MenuItem>
                <asp:MenuItem
            NavigateUrl="/RewriteTestVB/DoveSiamo"
            Text="Dove Siamo" Value="Dove Siamo">
                </asp:MenuItem>
                <asp:MenuItem
            NavigateUrl="/RewriteTestVB/Contattaci"
                Text="Contatti" Value="Contatti">
                </asp:MenuItem>
        </asp:MenuItem>
        <asp:MenuItem
        NavigateUrl="/RewriteTestVB/categorie.aspx"
```

WEB.CONFIG

Il web.config è un file di configurazione delle applicazioni .Net. E' un file XML diviso in sezioni, ciascuna della quali ha un compito ben preciso. Tra le principali ricordiamo <appsettings> in cui inseriamo delle variabili di configurazione che potranno essere lette a runtime.
E' possibile accedere in modo programmatico alle parti del web.config utilizzando il namespace System.Configuration.

```
Text="Prodotti" Value="Prodotti">

</asp:MenuItem>
</Items>
<StaticHoverStyle BackColor="#284E98"
    ForeColor="White" />
</asp:Menu>
```

Il controllo contiene la lista dei link “statici”.
Creiamo i seguenti file:

- Default.aspx
- Categorie.aspx
- SottoCategorie.aspx
- Prodotti.aspx

Il file Default.aspx verrà utilizzato per visualizzare le pagine di presentazione della nostra società. I contenuti saranno letti dalla tabella “Contents” del database ed il rewrite degli url sarà fatto grazie alla sezione “simple_rules” del web.config.
Vediamo il codice della pagina:

```
Dim tab As String = ""
If Not Request.QueryString("tab") Is Nothing Then
        tab = Request.QueryString("tab")
Else
        tab = "Home"
End If
Dim Conn As SqlConnection = New
    SqlConnection(ConfigurationManager.ConnectionStrin
        gs("ConnectionString").ConnectionString)
        Try
        Conn.Open()
                Dim Sql As String = "select *
                    from contents where
                con_short_name=@con_short_name"
                Dim Comm As SqlCommand = New
                        SqlCommand(Sql, Conn)
        Comm.Parameters.Add(New
            SqlParameter("@con_short_name", tab))
        Dim reader As SqlDataReader =
                    Comm.ExecuteReader()
        Try
                If reader.Read() Then
                        lbl_titolo.Text =
                    reader("con_name").ToString()
                        Page.Title =
                    reader("con_name").ToString()
                        lbl_page_html.Text =
                    reader("con_content").ToString()
                End If
        Finally
                reader.Close()
        End Try
Finally
        Conn.Close()
End Try
```

REGULAR EXPRESSIONS

Una regular expression è una stringa che, utilizzando dei caratteri particolari, riesce a descrivere un set di stringhe. Ad esempio l’espressione ^\d{5}$, rappresenta un numero di 5 cifre (come un codice postale). In .Net c’è un namespace System.Text.RegularExpressions che contiene le classi per definire ed utilizzare le regular expressions. Le classi principali sono Regex, che rappresenta una espressione regolare, e la classe Match, che rappresenta il risultato del match di una regular expression.

è molto semplice: legge dalla QueryString il valore di tab e carica i contenuti dal database. Fin qui niente di strano... il bello è che per richiamare la pagina “chi siamo” non occorre linkare /rewritetestvb/deafult.asxp?tab=chisiamo, ma è sufficiente scrivere /rewritetestvb/chisiamo e così per le altre pagine dell’area istituzionale.
I link sono infatti

- Home Page -> /RewriteTestVB/Home/
- Chi Siamo -> /RewriteTestVB/ChiSiamo/
- Dove Siamo -> /RewriteTestVB/DoveSiamo/
- Contatti -> /RewriteTestVB/Contatti

Passiamo alla parte prodotti.
Le pagine in gioco sono:

- **Categorie.aspx**: visualizza le categorie dei prodotti.
- **SottoCategorie.aspx**: visualizza le sottocategorie di una categoria scelta.
- **Prodotti.aspx**: visualizza i prodotti di una sottocategoria.

Le regole per la riscrittura degli url sono quelle della sezione “regex_rules”.
Iniziamo con le categorie: la pagina in questione è Categorie.aspx che contiene, oltre al menu di navigazione, un GridView su cui vengono caricate le categorie prodotti. Il codice completo della pagina aspx ed il relativo vb, lo trovate nel cd allegato.
E’ utile analizzare il codice che crea i link alla pagina delle sottocategorie:

```
Protected Sub grd_categorie_RowDataBound(ByVal
        sender As Object, ByVal e As
    System.Web.UI.WebControls.GridViewRowEventArgs)
        Handles grd_categorie.RowDataBound
        If e.Row.RowType =
            DataControlRowType.DataRow Then
            Dim lbl_cat_short_name As Label =
        CType(e.Row.Cells(0).FindControl("lbl_cat_
                    short_name"), Label)
            Dim anc_sca As HtmlAnchor =
```

```
        CType(e.Row.Cells(1).FindControl("anc_sca"),
                                            HtmlAnchor)
                  anc_sca.HRef =
    "../Sottocategorie/" + lbl_cat_short_name.Text + "/"
            End If
End Sub
```

Non fa altro che leggere il nome della categoria e creare dei link del tipo **Sottocategorie/Nome_Categoria**; sarà il nostro modulo "rewrite" ad occuparsi di riscrivere l'url come **Sottocategorie.aspx?cat= Nome_Categoria**

La pagina Sottocategorie.aspx funziona allo stesso modo della pagina Categorie.aspx, e contiene una serie di link (anche questi riscritti dal nostro modulo) alla pagina Prodotti.aspx .

Vediamo, infine la pagina dei prodotti. Il codice completo si trova nel CD allegato.

La pagina contiene una Label in cui scriveremo categoria e sottocategoria corrente ed un GridView in cui andremo a caricare la lista dei prodotti.

Nell'evento Page_Load leggiamo categorie e sottocategoria dalla QueryString e lanciamo i tre metodi sotto riportati:

```
Private Function getCategory(ByVal Conn As
            SqlConnection, ByVal cat As String) As String
        Dim Sql As String = "select cat_name from
                Categories where cat_short_name=@cat"
        Dim Comm As SqlCommand = New
                            SqlCommand(Sql, Conn)
        Comm.Parameters.Add(New
                        SqlParameter("@cat", cat))
        Return
                Convert.ToString(Comm.ExecuteScalar())
End Function
Private Function getSubCategory(ByVal Conn As
            SqlConnection, ByVal scat As String) As String
        Dim Sql As String = "select sca_name from
            SubCategories where sca_short_name=@scat"
        Dim Comm As SqlCommand = New
                            SqlCommand(Sql, Conn)
        Comm.Parameters.Add(New
                        SqlParameter("@scat", scat))
        Return
                Convert.ToString(Comm.ExecuteScalar())
End Function
Private Sub FillProductsGrid(ByVal Conn As
        SqlConnection, ByVal cat As String, ByVal scat As
                                                String)
        Dim Sql As String = _
" SELECT" & _
                    "   pro_name Nome, pro_model
                                    Modello, " & _
"   pro_description Descrizione, pro_price Prezzo" & _
                    " FROM Products" & _
                    " INNER JOIN Categories" & _
                    "   ON pro_cat_id = cat_id" & _
                    " INNER JOIN SubCategories" & _
                    "   ON pro_sca_id = sca_id" & _
                    " WHERE" & _
                    "   cat_short_name = @cat" & _
                    "   AND sca_short_name =
                                            @scat"
        Dim Comm As SqlCommand = New
                            SqlCommand(Sql, Conn)
        Comm.Parameters.Add(New
                        SqlParameter("@cat", cat))
        Comm.Parameters.Add(New
                        SqlParameter("@scat", scat))
        Dim myDataAdapter As SqlDataAdapter =
                        New SqlDataAdapter(Comm)
        Dim table As DataTable = New DataTable()
        myDataAdapter.Fill(table)
        myDataAdapter.Dispose()
        grd_prodotti.DataSource = table
        grd_prodotti.DataBind()
End Sub
```

getCategory restituisce il nome della categoria **getSubCategory** restituisce il nome della sottocategoria **FillProductsGrid** riempie la griglia con la lista dei prodotti. Se eseguiamo il progetto e clicchiamo su qualche link, possiamo notare come sulla barra degli indirizzi vengano sempre visualizzati gli indirizzi "user friendly" invece di quelli reali.

CONCLUSIONI

Con ASP 3 l'unico modo per realizzare l'url rewriting era scrivere un filtro ISAPI ed installarlo su IIS; una soluzione spesso irrealizzabile soprattutto se non si aveva accesso alla configurazione del server. Con .Net si hanno molte possibilità in più. La possibilità di utilizzare moduli anche a livello della singola applicazione consente di avere una portabilità maggiore delle proprie applicazioni rendendole meno dipendenti dalla configurazione del server su cui dovranno funzionare.

Carmelo Scuderi

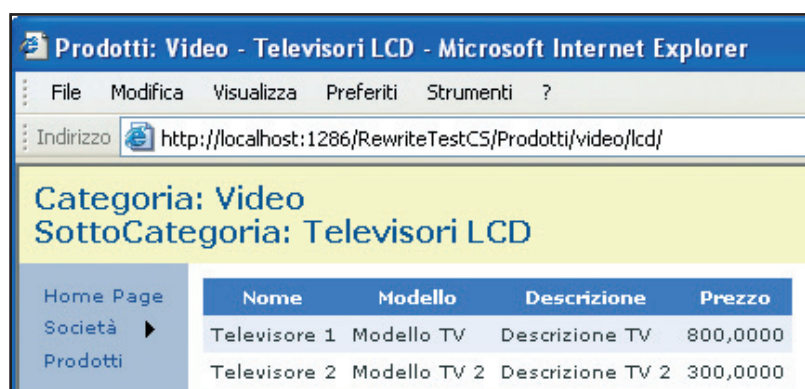
Fig. 4: **La pagina dei prodotti**

SUL WEB

Apache mod_rewrite
http://httpd.apache.org/docs/1.3/mod/mod_rewrite.html

Sulle regular expressions
http://en.wikipedia.org/wiki/Regular_expression
http://regexlib.com/

L'AUTORE

Carmelo Scuderi è ingegnere informatico. Si occupa di sviluppo software in ambiente .Net per una società di informatica di Milano. Gestisce un sito ricco di script e manuali per chi si affaccia al mondo della programmazione web (www.morpheusweb.it).

JAVASCRIPT E LE REGULAR EXPRESSION

IN QUESTO ARTICOLO APPRENDERETE COSA SONO LE ESPRESSIONI REGOLARI E COME UTILIZZARLE IN JAVASCRIPT. VEDREMO COME SARÀ POSSIBILE VALIDARE, LATO CLIENT, E-MAIL E QUANT'ALTRO E COME ESTRARRE PARTI DI STRINGHE

Un'espressione regolare (regex per gli amici) è una stringa che, utilizzando una speciale sintassi, identifica occorrenze di sottostringhe all'interno di una sequenza di caratteri. Per chiarire facciamo subito un esempio. Supponiamo che da una pagina HTML vogliamo estrarre tutti gli script contenuti nella stessa. La figura 1 dovrebbe chiarire il concetto. Sicuramente si può raggiungere questo risultato senza utilizzare le regex. Basta scrivere decine di righe di codice che usano **charAt**, **indexOf** e **substr** ed il gioco è fatto. Se invece conoscete le regex potete ottenere lo stesso risultato utilizzando una riga di codice! Interessante vero?

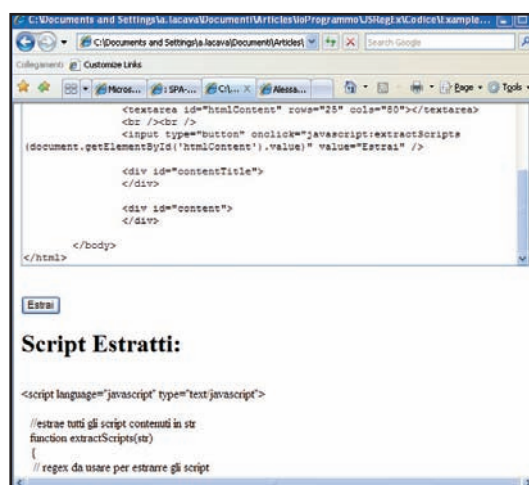



Fig. 1: Screenshot dell'applicazione

Alla fine dell'articolo vedremo come sviluppare, per l'appunto, una pagina Web che, inserendo il codice HTML in una textarea, estrarrà tutti gli script contenuti nello stesso. Questa si rivelerà molto utile nel caso in cui volessimo "sniffare" gli script utilizzati da una pagina Web senza andare a trovarli "a mano" in mezzo al resto del codice.
Bisogna dire che i concetti esposti in questo articolo, riguardo alle espressioni regolari, sono applicabili, apportando qualche piccolo accorgimento, a tutti quei linguaggi che supportano questa potente feature. Per i nostri scopi utilizzeremo JavaScript ce nella sua universalità ci consente di illustrare la tecnica senza agganciarci alle peculiarità di linguaggi più asettici

REGULAR EXPRESSION E JAVASCRIPT

JavaScript supporta le regex attraverso la classe RegExp. Vi sono due versioni del costruttore di questa classe; una che accetta un solo parametro e un'altra due. Facciamo subito qualche esempio.

```
var str = "Mosca è una bella città";
var regex = new RegExp("mosca");
if(regex.test(str))
        alert("Trovata occorrenza della parola
                                        mosca");
else
        alert("Non è stata trovata alcuna
                occorrenza della parola mosca");
```

Eseguendo questo script noterete che verrà visualizzato il secondo alert, ovvero la parola non viene trovata. Questo perché, per default, le espressioni regolari sono case-sensitive. Notare, inoltre, l'uso del metodo test della classe RegExp. Tale metodo, restituisce true se e solo se il pattern trova almeno un match. Se, invece, si vuole che la ricerca non faccia differenza tra maiuscole e minuscole, si può utilizzare la seconda forma del costruttore. Infatti, sarebbe bastato istanziare l'oggetto nel seguente modo:

```
var regex = new RegExp("mosca", "i");
```

e sarebbe stato visualizzato il primo alert. La *i*

Javascript

indica di fare una ricerca di tipo case-insensitive. In questo esempio la regex era costituita da una semplice stringa: "mosca". Come vedremo, però, la potenza delle espressioni regolari risiede nel fatto che è possibile utilizzare numerosi simboli per soddisfare i più svariati bisogni. Ad esempio, la seguente regex identifica tutte le parole che iniziano per m, finiscono per a e non fa differenze tra maiuscole e minuscole:

```
var regex = new RegExp("m\\w*a", "i");
```

Essa rappresenta, quindi, le seguenti parole: mosca, mora, marina, menta…ma anche Marina, MARta, ma, MA, mA ecc. Questo perché \w identifica la classe dei caratteri alfanumerici, incluso l'underscore. Il quantificatore * significa zero o più di ciò che lo precede; nel nostro caso zero o più caratteri alfanumerici o underscore.
Non preoccupatevi però, spiegheremo più avanti le classi ed i quantificatori.
A parte test, esiste un altro metodo della classe RegExp molto potente: exec. Tale metodo restituisce un array di stringhe contenente, all'indice 0, l'espressione trovata e nei seguenti indici gli eventuali gruppi (che vedremo in seguito). Se non trova nessun match, exec restituisce false. Rifacendoci all'esempio precedente:

```
var str = "mosca, mora, marina, Alessandro,
     menta, ioProgrammo, Marina, MARta, ma, MA,
                                             mA";
var regex = new RegExp("m\\w*a", "i");
var results = regex.exec(str);
alert(results.toString());
```

Eseguendo quest'esempio noterete che viene visualizzata solo la parola mosca. Perché? Il motivo risiede nel fatto che, per default, la regex identifica solo la prima occorrenza. Se si vuole indicare di trovare tutte le occorrenze, bisogna usare un'altra opzione. Tale opzione è g e sta per global. Modificando l'espressione regolare nel modo seguente:

```
var regex = new RegExp("m\\w*a", "gi");
```

e ciclando attraverso i risultati trovati:

```
var ret = [];
for(var results = regex.exec(str); results != null;
                        results = regex.exec(str))
{
  ret.push(results);
}
alert(ret.toString());
```

noteremo che sarà visualizzato l'intero array di match trovati. È necessario ciclare attraverso i risultati, poiché altrimenti sarebbe visualizzato di nuovo solo la prima occorrenza.
Alternativamente ad exec è possibile utilizzare il metodo match della classe String che permette di ottenere direttamente l'array contenente tutti i risultati. Non vi è quindi la necessità di ciclare per recuperare tutte le occorrenze. Ad esempio:

```
var str = "mosca, mora, marina, Alessandro,
     menta, ioProgrammo, Marina, MARta, ma, MA,
                                             mA";
var regex = new RegExp("m\\w*a", "gi");
var results = str.match(regex);
alert(results.toString());
```

Nel caso in cui non si ha la necessità di utilizzare gruppi, consiglio di usare il metodo match. Da notare che tale metodo appartiene alla classe String, non a RegExp, ed accetta come parametro una regex. Vi sono altri due metodi molto utili della class String che accettano come parametro un'espressione regolare. Essi sono replace e split. Il primo sostituisce tutte le occorrenze della prima stringa con la seconda. Ad esempio:

```
var str = "mosca, mora, marina, Alessandro,
     menta, ioProgrammo, Marina, MARta, ma, MA,
                                             mA";
var regex = new RegExp("m\\w*a", "gi");
var results = str.replace(regex, "sostituita");
alert(results.toString());
```

Il risultato di questo script è visibile in figura

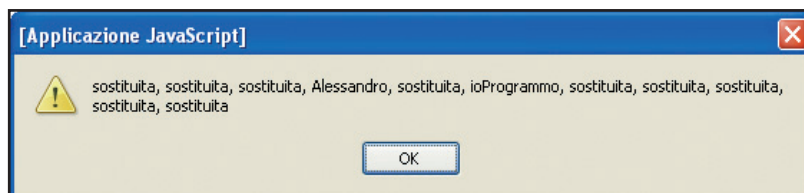



Fig. 2: *Risultato dell'esecuzione dello script*

PATTERN E MATCH

È proprio vero! Alcune parole tecniche inglesi non sono direttamente traducibili in italiano. In quest'articolo uso spesso le parole pattern e match. Per pattern s'intende la regex vera e propria, ossia la stringa identificante l'espressione regolare, come ad esempio 02-\\d{5}. Per match s'intende che una stringa soddisfa la nostra regex. Ad esempio 02-12387 soddisfa il pattern 02-\\d{5}.

2. Come si può vedere, sono state sostituite tutte le occorrenze trovate. Il metodo split, invece, divide una stringa in sottostringhe e le ritorna come un array. Ad esempio:

```
var str = “primo elemento-secondo
                    elemento-terzo elemento”;
var regex = new RegExp(“-”, “g”);
var results = str.split(regex);
alert(results.toString());
```

In questo esempio viene estratto un array contenente gli elementi che, in str, sono separati dal segno -. Il metodo split è molto utile nel caso in cui abbiamo a che fare con CSV.

METACARATTERI, CLASSI DI CARATTERI E QUANTIFICATORI

I metacaratteri, assieme a classi e quantificatori, sono i pilastri fondamentali delle espressioni regolari e questo a prescindere dal linguaggio di programmazione in uso.
Un metacarattere non è nient'altro che un carattere che possiede un particolare significato all'interno di un'espressione regolare. I metacaratteri sono:

```
( [ { \ ^ $ | ) ? * + .
```

Scopriremo il significato di questi caratteri nel prosieguo dell'articolo. Per ora vi basti sapere che, se volete utilizzare uno di questi caratteri all'interno di una regex, dovrete farlo passare “inosservato” utilizzando il carattere di escape, cioè il back slash \. Ad esempio, se nello script precedente avessimo avuto stringhe separate dal carattere pipe, |, invece che dal segno -, allora avremmo dovuto utilizzare l'escape per far funzionare le cose.

```
var str = “primo elemento|secondo
                    elemento|terzo elemento”;
var regex = new RegExp(“\\|”, “g”);
var results = str.split(regex);
alert(results.toString());
```

Classe	Identifica	Equivalente
.	Qualsiasi carattere tranne \n e \r	[^\n\r]
\d	Qualsiasi numero	[0-9]
\D	Qualsiasi carattere tranne numeri	[^0-9]
\s	Uno spazio	[\n\r\t\f\x0B]
\S	Qualsiasi carattere tranne uno spazio	[^ \n\r\t\f\x0B]
\w	Qualsiasi carattere alfanumerico e l'underscore	[a-zA-Z0-9_]
\W	La negazione di \w	[^a-zA-Z0-9_]

Tabella 1: Classi predefinite

È da notare una cosa importante, ovvero l'utilizzo del doppio back slash, \\. Questo è necessario perché se guardate i metacaratteri troverete che tra essi è presente anche il simbolo \. Quindi bisogna utilizzare il carattere di escape per far passare inosservato l'escape stesso.
Oltre ai metacaratteri bisogna considerare altri speciali caratteri predefiniti come nuova linea o tab, rispettivamente \n e \t. Se dovete utilizzare tali sequenze all'interno delle regex, bisogna utilizzare l'escape.
Passiamo ora alle classi. Una classe denota un insieme di caratteri. Le classi si indicano tra parentesi quadre. Ad esempio,

```
var str = “Mosca, Tosca, Ciao Mondo”;
var regex = new RegExp(“[mt]osca”, “gi”);
var results = str.match(regex);
alert(results.toString());
```

Il precedente esempio trova due match: Mosca e Tosca. In sostanza quella regex significa: trovami tutte (l'opzione g) le occorrenze di mosca e/o tosca (la classe [mt]) a prescindere dalle maiuscole e minuscole (l'opzione i). Vi consiglio di iniziare a ragionare in questo modo, ovvero a tradurre in parole le espressioni regolari che incontrate.
Questo vi aiuterà a prendere dimestichezza con le stesse.
L'uso delle classi non si limita a casi semplici. Potete indicare un range di caratteri. Ad esempio, la regex:

```
var regex = new RegExp(“[a-z]osca”, “gi”);
```

identifica tutte le parole che iniziano per qualsiasi carattere e terminano per osca. È anche possibile combinare più classi assieme:

```
var regex = new RegExp(“[m-z1-9]osca”, “gi”);
```

Quest'esempio “matcha” tutte le parole che iniziano per m, n,…z oppure 1, 2,…,9 e terminano in osca.
È possibile, inoltre, utilizzare le negazioni. Ad esempio [^a-d] identifica qualsiasi carattere tranne a, b, c, d. In definitiva, le classi di caratteri sono molto utili e potenti.
I lettori dotati di buona memoria ricorderanno che all'inizio ho utilizzato i caratteri \w

per identificare qualsiasi carattere alfanumerico, incluso l'underscore. Quella usata non è nient'altro che una classe predefinita. In pratica, è una sorta di scorciatoia. La **tabella 1** illustra tutte le classi predefinite e ne descrive il significato.
Passiamo ora ai quantificatori. Come dice la parola stessa, i **quantificatori** ci permettono di **quantificare** il numero di occorrenze di un insieme di caratteri. Se ricorderete, a parte \w, all'inizio dell'articolo, ho utilizzato anche il simbolo *. Tale simbolo significa zero o più occorrenze di ciò che lo precede. La **tabella 2** illustra i diversi quantificatori ed il loro significato.
Facciamo un esempio. Supponiamo che in una stringa siano contenuti dei numeri di telefono preceduti dal prefisso e di voler estrarre solo i numeri di Milano, in altre parole quelli che hanno 02 come prefisso. Ecco il codice che fa al caso nostro:

```
var str = "Il mio numero è 02-123456789,
mentre il numero di Tizio è 06-12309876.
Il mio amico  Caio invece ha il numero
           02-987654321. Notare
     che se 02- non è seguito da almeno una cifra,
      come in questo caso, non verrà considerato";
var regex = new RegExp("02-\\d+", "gi");
var results = str.match(regex);
alert(results.toString());
```

Voglio far notare la differenza tra {n} e {n,} come quantificatori. Se, ad esempio, avessimo usato:

```
var regex = new RegExp("02-\\d{5}", "gi");
```

il risultato sarebbe stato 02-12345 e 02-98765. Questo perché quella regex significa: "Trova le occorrenze che iniziano con 02, seguito dal segno - e poi da 5 cifre. Vale a dire, trovate le 5 cifre non continuare a cercare perché a me interessano solo quelle". Se invece usassimo:

```
var regex = new RegExp("02-\\d{5,}", "gi");
```

il risultato sarebbe 02-123456789 e 02-987654321. Questo perché la regex significa: "Trova le occorrenze che iniziano con 02, seguito dal segno - e poi da almeno 5 cifre. Cioè, se trovi 5 cifre non ti fermare, ma va avanti finché continui a trovare numeri".
Guardate ora il seguente codice:

```
var str = "Questa è una frase. Questa è un'altra
                                        frase.";
var regex = new RegExp("Questa.+\\.", "g");
var results = str.match(regex);
alert(results.toString());
```

se vi aspettate che il risultato sia un array contenente due elementi costituiti da "Questa è una frase." e "Questa è un'altra frase." allora vi sbagliate. Il risultato è un array costituito dall'unico elemento "Questa è una frase. Questa è un'altra frase.". Provate invece a sostituire la regex vista con la seguente:

```
var regex = new RegExp("Questa.+?\\.", "g");
```

È quasi uguale alla precedente se non per il fatto di avere il simbolo ? che segue +. Se provate a lanciare lo script ora noterete che funziona come vi aspettavate. Perché? La risposta risiede nel concetto di quantificatori greedy, reluctant e possessive.

QUANTIFICATORI GREEDY, RELUCTANT E POSSESSIVE

I quantificatori visti fino a questo momento sono di tipo greedy (ingordi). Un quantificatore "ingordo" inizia a cercare un match sull'intera stringa. Se lo trova ha terminato altrimenti inizia a togliere un carattere alla volta, comportandosi così da ingordo! Detto in altre parole, un quantificatore greedy cerca di trovare un match utilizzando il maggior numero di caratteri possibili. Nell'esempio precedente, infatti, avendo la regex:

```
var regex = new RegExp("Questa.+\\.", "g");
```

e la stringa:

```
var str = "Questa è una frase. Questa è un'altra
                                        frase.";
```

il match trovato era: "Questa è una frase. Questa è un'altra frase.". Questo perché + è greedy e trova, nell'intera stringa, il match desiderato. Infatti l'intera stringa inizia per

Quantificatore	Descrizione
*	Zero o più occorrenze
+	Una o più occorrenze
?	Zero o un'occorrenza
{n}	Esattamente n occorrenze
{n,}	Minino n occorrenze
{n,m}	Minimo n e massimo m occorrenze

Tabella 2: Quantificatori

"Questa", è seguita da uno o più caratteri e termina con un punto.
I quantificatori di tipo reluctant (riluttanti) si comportano in modo opposto. Sono, come dire, più "educati". Infatti, cercano di trovare un match col minor numero di caratteri possibili. In pratica, un quantificatore reluctant inizia col primo carattere e ne aggiunge uno per volta finché trova un match. Infatti, nel nostro esempio:

```
var regex = new RegExp("Questa.+?\\.", "g");
var str = "Questa è una frase. Questa è un'altra
                                     frase.";
```

vengono trovati due match grazie all'uso del quantificatore +? che, come detto, è reluctant. Un quantificatore di tipo possessive (possessivo) controlla se l'intera stringa "matcha" con il pattern dato. Se non trova il match non fa ulteriori controlli. Per indicare che un quantificatore è possessive basta farlo seguire dal simbolo +. Non entreremo nel dettaglio dei quantificatori possessive dato che non sono molto supportati dagli attuali browser. Inoltre, simulare un quantificatore possessive utilizzando il greedy non è molto difficile. In **tabella 3** sono illustrati i tre tipi di quantificatori. Potete tranquillamente consultarla per realizzare le combinazioni che caso per caso risolvono il vostro problema

Greedy	Reluctant	Possessive	Descrizione
*	*?	*+	Zero o più occorrenze
+	+?	++	Una o più occorrenze
?	??	?+	Zero o un'occorrenza
{n}	{n}?	{n}+	Esattamente n occorrenze
{n,}	{n,}?	{n,}+	Minino n occorrenze
{n,m}	{n,m}?	{n,m}+	Minimo n e massimo m occorrenze

Tabella 3: Tipi di quantificatori

SINTASSI PERL-LIKE

Perl è stato, probabilmente, il primo linguaggio di programmazione ad implementare le espressioni regolari. JavaScript introdusse il supporto alle regex ancor prima di Java, il quale le introdusse dalla versione 1.4. Dato che JavaScript si basò sul linguaggio Perl per l'implementazione delle regex, ne mantenne anche la sintassi. È, infatti, possibile esprimere un'espressione regolare in modo letterale utilizzando la sintassi: /02-\d{5}/g. Ecco un esempio:

```
...
var regex = /02-\d{5}/g;
var results = str.match(regex);
```

Da notare che, usando la sintassi Perl-like, non vi è la necessità dell'escape per le classi predefinite.

I GRUPPI

Arrivati a questo punto sorge un problemino. Supponiamo vi venga in mente di scrivere un'espressione regolare che, data una pagina HTML, estragga il contenuto dell'attributo src di tutti i tag img. In altre parole, gli URL di tutte le immagini presenti nella pagina.
Con le conoscenze acquisite fino a questo momento già potete ottenere questo risultato. Infatti, potete scrivere una regex per estrarre tutti i tag img e poi un'altra che su ogni tag estratto trovi il contenuto di src. Vedremo, in ogni modo, che le espressioni regolari ci vengono incontro e ci permettono di fare lo stesso lavoro con una singola regex. Il concetto dietro il quale si cela questa "magia" risponde al nome di raggruppamento. È infatti possibile dividere una regex in vari gruppi. Un gruppo è racchiuso all'interno delle parentesi tonde. Il gruppo 0 corrisponde all'intera regex, il gruppo 1 alla prima coppia di parentesi tonde, il 2 alla seconda e così via. Scriviamo subito l'esempio che, da una pagina HTML estrae gli URL delle immagini presenti in essa.

```
var str = "<html><head></head><body><img
                       src='./images/test1.jpg' />
                       <!--Altro codice HTML --->
                    <img src='./images/test2.jpg'
                              /></body></html>";
var regex = new
RegExp("<\\s*img.*?src\\s*=\\s*(\"|')(.+?)\\1.*?
                                        >", "gi");
var ret = [];
for(var results = regex.exec(str); results != null;
                   results = regex.exec(str))
{
  ret.push(results[2]);
}
alert(ret.toString());
```

Lanciando questo script noterete che saranno visualizzati i due URL relativi, in altre parole "./images/test1.jpg" e "./images/test2.jpg". Ricordo che il metodo exec della classe RegExp estrae i risultati mettendo l'intero match, cioè il primo gruppo (rappresentato dall'intera regex), all'indice 0 e i vari gruppi negli indici successivi. Il gruppo che interessa a noi è il terzo, ossia (.+?). Notare l'uso del simbolo pipe, |, all'interno di (\"|'). In pratica svolge la funzione di un OR. Significa: "Vanno bene sia i doppi sia il singolo apice". Osservare, inoltre, l'uso dei quantificatori di tipo reluctant.
Per imparare bene le espressioni regolari bisogna sperimentare e sperimentare e…OK

avete capito. Provate, ad esempio, ad utilizzare quantificatori di tipo greedy nel medesimo codice e osservate i risultati.
Osservando l'espressione regolare precedente, emerge l'uso di qualcosa che non avevamo visto prima, ovvero il codice \1. Questa tecnica è chiamata "backreferencing". In pratica, essa permette di riferirsi ad una stringa che è stata "catturata" in qualche gruppo precedente. Nel nostro caso \1 significa: "Quello che hai trovato utilizzando il gruppo 1, ossia (\"|')". In altre parole, se ha trovato un doppio apice allora \1 significa doppio apice, mentre se ha trovato un singolo apice \1 sarà sostituito con un singolo apice. Troverete la tecnica di backreferencing molto utile in svariate situazioni.

L'APPLICAZIONE DI ESEMPIO

Come detto all'inizio del presente articolo, vedremo ora come scrivere un pagina Web, contenente una textarea e un bottone, in grado di estrarre tutti gli script contenuti nel codice HTML "incollato" nella textarea stessa. La figura 1 ne illustra il banale layout. Quello che segue è il codice della parte "core":

```
//estrae tutti gli script contenuti in str
function extractScripts(str)
{
// regex da usare per estrarre gli script
var reScript = new
                RegExp("<\\s*script.*?>(.|\\n|\\r)*?
                       <\\s*/\\s*script\\s*>", "gi");
// estrae gli script mettendoli in un array
var scriptArray = str.match(reScript);
// costrisce la stringa da visualizzare
var html = buildString(scriptArray);
// la visualizza
display(html);
}
```

L'unica cosa degna di nota è, ovviamente, l'espressione regolare usata. In particolare, voglio chiarire la parte (.|\\n|\\r)*?. In sostanza vuol dire: "Qualsiasi carattere o un carattere di nuova linea (\n) o un ritorno a capo (\r). Il quantificatore indica di prendere zero o più occorrenze del gruppo che lo precede.". L'uso di \n e \r è necessario in quanto, nel caso in cui lo script si estenda su più righe, non verrebbero prese in considerazione dato che il punto (qualsiasi carattere) non contempla i ritorni a capo e le nuove linee. Sulle funzioni buildString e display non vi è molto da dire. La prima costruisce la stringa che rappresenta gli script estratti. La seconda la visualizza.
Da notare che, anche se lo script è in un file esterno, la nostra applicazioncina ci visualizza il percorso a cui punta. Se volessimo vedere il sorgente del file ci basterebbe puntare il nostro browser a quel percorso.

CONCLUSIONI

Le espressioni regolari sono molto potenti. La cosa più importante da capire, comunque, è che i concetti esposti in quest'articolo sono applicabili a qualsiasi linguaggio di programmazione che supporta le regex. Infatti, concetti quali: metacaratteri, classi, quantificatori greedy, reluctant e possessive, gruppi e backreference sono completamente generali e, apportando le opportune modifiche, potete utilizzare questi strumenti in ogni linguaggio "regex-compliant".
Vi è da dire tuttavia, che ci sono interi libri dedicati alle espressioni regolari e, ovviamente, in quest'articolo non abbiamo potuto coprire tutti i concetti ad esse relativi. Gli strumenti analizzati, in ogni modo, sono quelli di maggiore utilizzo e vi permetteranno di risolvere la stragrande maggioranza dei problemi. Un'ultima nota: nell'esempio che tratta l'estrazione delle immagini non ho tenuto conto della possibilità che il tag img si estenda su più righe, come invece ho fatto per l'estrazione degli script. Per non escludere questa possibilità dovrete modificare leggermente la regex in questione. Ovviamente tale compito è lasciato per esercizio. La potenza delle regex si comprende appieno ad esempio in problemi di normalizzazione dei database. Grazie alla loro fkessibilità è infatti possibile ottenere soluzioni efficienti in tempi decisamente brevi.

Alessandro Lacava

VALIDAZIONE DI E-MAIL

Le regex sono spesso utilizzate in fase di validazione dei campi. Ad esempio, validare un indirizzo e-mail è molto semplice. A tal proposito, potete utilizzare la seguente espressione regolare: /^(\w+\.?)*\w+@(\w+\.{1})+\w+$/. I caratteri ^ e $ indicano, rispettivamente, l'inizio e la fine della stringa. Il loro uso è necessario perché se una persona inserisse "prova@nomesito.com, ciao", verrebbe interpretato erroneamente come indirizzo valido. Utilizzando ^ e $ specifichiamo che le espressioni valide sono quelle che rispettano la regex e non sono precedute o seguite da altri caratteri.

IL BROWSER SUL DESKTOP

PROGETTARE UN'INTERFACCIA È SPESSO PIÙ FACILE PER IL WEB CHE PER LE APPLICAZIONI STANDALONE. ALLORA PERCHÈ NON UNIRE LE DUE COSE PROGETTANDO SOFTWARE CHE GIRA SUL TUO COMPUTER MA HA UNA GRAFICA HTML?

Utilizzare una finestra di Internet Explorer all'interno di un'applicazione desktop non è certo una novità per i programmatori Visual Basic.
Fin da Visual Basic 6, infatti era possibile disporre, come oggetto COM, del controllo WebBrowser, il cui uso era spesso quello di mostrare file HTML di help o pagine web come possiamo vedere in figura 1.

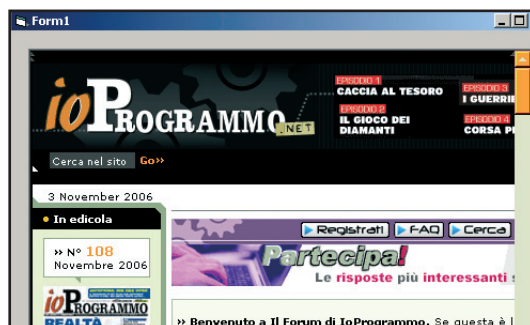



Fig. 1: **un controllo WebBrowser in una form VB6**

Tale controllo era anche disponibile per i programmatori .NET 1.0 e 1.1 attraverso il wrapper dello stesso oggetto COM, l'utilizzo della finestra di Internet Explorer, però è rimasto sostanzialmente lo stesso.
Il colloquio tra applicazione desktop e quanto avviene dentro il controllo WebBrowser non era impossibile, ma certo non era impresa di poco conto.
Con .NET 2.0 le cose sono cambiate radicalmente, innanzitutto abbiamo adesso nella barra degli strumenti di Visual Studio 2005 un controllo nuovo di zecca chiamato, appunto, WebBrowser.
Intendiamoci, alla base c'è sempre il solito oggetto COM che abbiamo visto in precedenza, ma è stato "incapsulato" nel codice managed in maniera magistrale.
In particolare il controllo Web Browser espone alcune proprietà veramente interessanti:

- Proprietà ObjectForScripting – che consente di esporre un oggetto al codice javascript della pagina contenuta.
- Proprietà DocumentText – che permette di caricare qualsiasi stringa HTML nella pagina
- Proprietà Document – che espone il documento caricato nella finestra come oggetto HTMLDocument per interagire in tutti modi possibili con il DOM sottostante.

Fig. 2: **il controllo WebBrowser nella Toolbar di VS 2005**

A COSA SERVE TUTTO QUESTO?

Certo che a questo punto vi starete domandando: "bello, ma a me ...?". Tuttavia, se ci pensate bene, le implicazioni di questi che sembrano piccoli miglioramenti sono in realtà veramente importanti.
Qual è l'attività più dispendiosa e meno produttiva di un programmatore di applicazioni desktop? Sono sicuro che anche voi risponderete: il disegno dell'interfaccia utente.
Quante ore perdete a disegnare sulle Forms bottoni, input box, combobox, tabelle ecc… Se poi dovete cambiare di posto a qualche elemento? Se dovete eliminare un bottone che magari è referenziato in venti posti diversi del codice?
Non parliamo poi di quelli che vogliono fare i "raffinati" e magari vorrebbero che in una textbox comparisse un link o che in una cella di tabella ci venisse un bel combobox ecc… Per loro comincia la "via crucis" dei controlli personalizzati che, spesso, risulta più dispendiosa, in termini di tempo perché magari il denaro è sempre lo stesso, dello sviluppo

dell'applicazione stessa.
Alla fine, parliamoci chiaro, molti programmatori utilizzano i controlli che mette a disposizione mamma Microsoft o, se proprio non ne possono fare a meno, comprano qualche controllo prodotto da terze parti
Il risultato è, spesso, un'interfaccia utente poco entusiamante, per niente intuitiva e spesso brutta.
Per contro, invece, qual è la tecnologia che permette la massima flessibilità in campo di disegno di interfacce utente: di certo l'HTML, specie da quando si è arricchito delle enormi possibilità offerte dai CSS e dal DOM manipolabile con Javascript.
Non c'è dubbio che, come interfaccia utente, un sito web di medio livello sia di molti punti superiore ai normali programmi desktop che si vedono in circolazione.
Permettere l'interazione tra applicazione Host e il controllo WebBrowser quindi ci consente di utilizzare, in tutto o in parte, una perfetta GUI web-like in modo semplice ed efficace.
Ma passiamo ad un esempio che ci fa subito entrare nel merito.

UN PO' DI PRATICA

Facciamo subito un esempio concreto. Ci viene commissionata un'applicazione che effettua una ricerca in un database SQL server, mostra i dati in una tabella con paginazione e consente l'esportazione dei dati in Excel.
Decidiamo che l'interfaccia utente sia tutta web-like. Quindi la logica di business risiederà nella nostra applicazione desktop, mentre il front-end sarà tutto HTML.
Per prima cosa avviamo un nuovo progetto Windows Application in Visual Studio 2005 (va bene anche Visual Studio Express).
Abbiamo deciso di gestire tutta l'interfaccia da HTML, quindi inseriamo una nuova Form senza barra del titolo (tipo splash screen per intenderci), inseriamo quindi due controlli Web: il primo, ancorato al top della form, verrà utilizzato come barra del titolo alternativa, l'altro, che riempie lo spazio rimanente, invece servirà ad ospitare tutta la GUI, come possiamo vedere in figura 3.

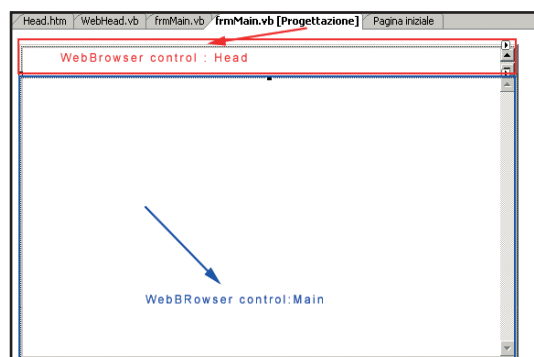



Fig. 3: ***i controlli WebBrowser sulla Form***

Prima di procedere oltre dobbiamo però ancora effettuare un'operazione importante: nelle proprietà del progetto, nella scheda "Applicazione" (la prima) clicchiamo sul bottone "informazioni assembly", nella finestra che apparirà selezioniamo quindi l'opzione "Rendi assembly visibile a COM" (vedi figura 4), questo ci consentira di interagire da codice con i controlli appena creati.

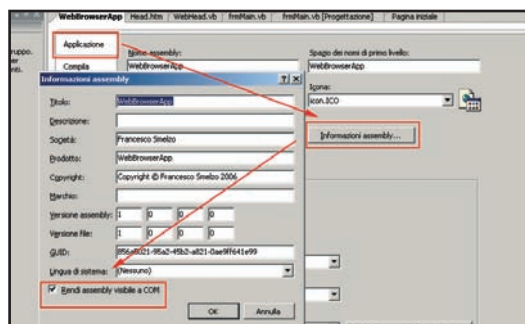



Fig. 4: ***opzioni del progetto***

LA BARRA DEL TITOLO PERSONALIZZATA

Lavoriamo adesso sul primo WebBrowser Control, che abbiamo chiamato Head, che servirà a rappresentare un control box personalizzato per la nostra applicazione, con i bottoni per minimizzare la finestra, ingrandirla e chiuderla.
In questa fase non ci cureremo della grafica, ma ci concentreremo sulle funzionalità.
Quindi, per prima cosa, creiamo una classe che espone al codice javascript del documento caricato nel WebBrowser Control le funzioni di base per gestire le operazioni con la finestra. Creiamo tale classe aggiungendo al progetto il seguente codice:

```
<Microsoft.VisualBasic.ComClass()> _
Public Class WebHead
    Dim ParentForm As Form
    Public Sub New(ByVal parentForm As Form)
        Me.ParentForm = parentForm
    End Sub
    Public Sub Minimize()
        Me.ParentForm.WindowState =
                                FormWindowState.Minimized
    End Sub
    Public Sub Maximize()
        Me.ParentForm.WindowState =
                                FormWindowState.Maximized
    End Sub
    Public Sub Normal()
        Me.ParentForm.WindowState =
                                FormWindowState.Normal
    End Sub
```

Una libreria offerta dalla Apple, QuickTime for Java, può essere usata per convertire file audio-video, adattandoli allo standard riconosciuto dal IPOD.
http://developer.apple.com/quicktime/qtjava/
http://www.onjava.com/pub/a/onjava/2003/05/14/qtj_reintro.html
http://www-128.ibm.com/developerworks/wireless/library/wi-mvideo/?ca=dgr-lnxw17CreateIPODvideo

```
    Public Sub Close()
        Me.ParentForm.Close()
    End Sub
    Public Function WindowState() As String
        Return
            Me.ParentForm.WindowState.ToString.ToLower
    End Function
End Class
```

Sul contenuto c'è poco da commentare, si tratta delle comuni funzioni di gestione di una Windows Form (che viene passata come argomento del costruttore New), da notare l'attributo Microsoft.VisualBasic.ComClass() che sta ad indicare che questa è una classe esposta a COM (e quindi anche al controllo WebBrowser).
Creiamo quindi il documento HTML Head.htm che andrà a popolare il WebBrowser Control aggiungendolo al progetto. È importante selezionare l'opzione "Copia" nelle proprietà del file alla voce "Copia nella directory di output" (figura 5), in tal modo in fase di compilazione anche i file HTML verranno copiati nella directory del programma.

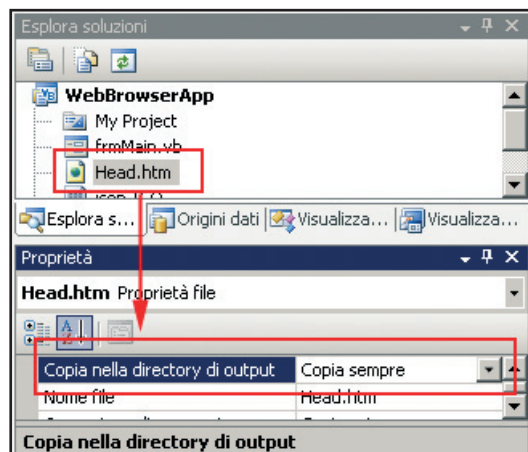



Fig. 5: Copia nella directory di output i file HTML

Prima di andare ad editare il file HTML però dobbiamo fare ancora una cosa nel codice della Window Form: dobbiamo instanziare un oggetto di tipo WebHead (la classe che abbiamo creato in precedenza) e definirlo come oggetto accessibile allo scripting per il WebBrowser Control Head.
Facciamo tutto questo nel codice della Form:

```
Public Class frmMain
    Private _HeadScriptingObject As WebHead
    Private ReadOnly Property HeadScriptingObject() As
                                                WebHead
        Get
            If _HeadScriptingObject Is Nothing Then
                _HeadScriptingObject = New WebHead(Me)
            End If
            Return _HeadScriptingObject
        End Get
    End Property
    Private Sub frmMain_Load(ByVal sender As
    System.Object, ByVal e As System.EventArgs) Handles
                                            MyBase.Load
Me.Head.Navigate(IO.Path.GetFullPath("Head.htm"))
        Me.Head.ObjectForScripting = HeadScriptingObject
    End Sub
End Class
```

In pratica abbiamo prima definito una proprietà che crea automaticamente un istanza di WebHead (con riferimento alla finestra corrente), poi abbiamo inserito, nell'evento Load della Form, il codice necessario ad aprire il file Head.htm nel controllo WebBrowser Head e assegnare a quest'ultimo l'istanza del tipo WebHead come oggetto di scripting.
A questo punto l'attenzione si sposta sul file HTML, che praticamente fa da "Client" (mentre l'applicazione fa da "Server"). L'oggetto con il quale Javascript può parlare con l'applicazione è window.external che, in questo caso, sarà proprio l'oggetto di tipo WebHead che abbiamo assegnato al WebBrowser Control.
Quindi, nel file Head.htm inseriremo una sezione script di questo tipo:

```
<script language="javascript" type="text/ecmascript">
var context=window.external;
function minimize(){ context.Minimize();}
function maximize(){ context.Maximize();}
function closeWindow(){ context.Close();}
function windowState(){ return context.WindowState();}
function normal(){ context.Normal();}
function switchState(caller) {
  if(windowState()=="normal") {
      maximize();
      caller.value="= ="
  }
    else{
      normal();
      caller.value="[]"
  }
}
</script>
```

In pratica abbiamo fatto un "wrapper" client delle funzioni esposte dall'oggetto di tipo WebHead rimappando l'oggetto window.external sulla variabile context per abbreviare il codice e inserendo una funzione di switch per il bottone che dovrà gestire l'ingrandimento e il ridimensionamento della finestra.
L'HTML vero e proprio sarà, per ora veramente minimo:

```
<body bgcolor="Background" text="white"
                            style="margin:0px">
<table width="100%" cellpadding="0" cellspacing="0">
```

```
    <tr>
        <td>Applicazione</td>
        <td align="right">
            <table cellpadding="0" cellspacing="1">
                <tr>
                    <td><input type="button"
                        onclick="minimize()" value="_" /></td>
                    <td><input type="button"
                onclick="switchState(this)" value="[]" /></td>
                    <td><input type="button"
                    onclick="closeWindow()" value="X" /></td>
                </tr>
            </table>
        </td>
    </tr>
</table>
</body>
```

A questo punto siamo pronti per la prima esecuzione, quindi premiamo il tasto F5 ottenendo la finestra che vediamo in figura 6.

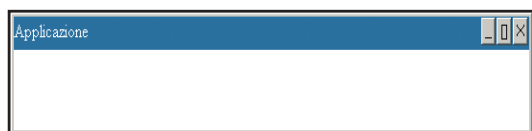



***Fig. 6:* La prima esecuzione**

Certo che il risultato non è niente di eccezionale, noterete però che le chiamate alle funzioni dell'applicazione Host funzionano perfettamente.
Lavoriamo quindi un po' sulla formattazione e gli stili CSS del file HTML ed otteniamo facilmente – e senza dover ricompilare nulla – il risultato di cui in figura 7.

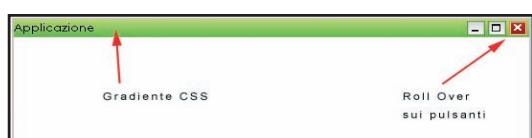



***Fig. 7:* La nostra barra del titolo migliorata**

Non ci dilunghiamo sugli stili applicati, basta dare un'occhiata ai sorgenti allegati al CD, c'è solo da sottolineare che tutte le risorse usate dal file HTML (icone,CSS,script esterni) devono essere inclusi nella directory del progetto e contrassegnati con l'opzione "copia nella directory di output".

LA CONNESSIONE AL DATABASE

Prima di costruire il front-end vero e proprio dobbiamo predisporre la nostra logica di business, quindi per prima cosa – nelle impostazioni del progetto – settiamo la nostra stringa di connessione in modo da poterla facilmente cambiare a runtime (**figura 8**).

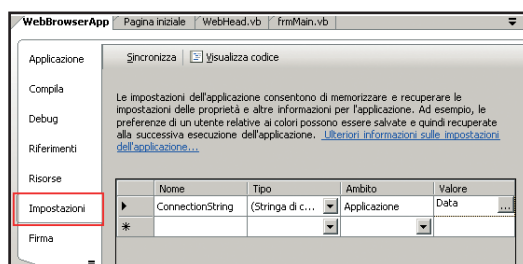



***Fig. 8:* Impostazione della connessione**

Il workflow delle chiamate sarà:

- Il codice HTML invia una richiesta con i parametri di selezione
- L'applicazione compone la query ed estrae i dati in XML, processa questi dati con un foglio di stile XSL e rinvia la stringa HTML alla pagina web
- Con i dati formattati la pagina web riempie un elemento DIV mostrando i risultati

Impostiamo quindi, nel nostro progetto, la classe che fa da interfaccia verso il database, da un lato e verso il codice Javascript dall'altro.
Come abbiamo fatto per il WebBrowser Control Head, anche per il WebBrowser Control Main creeremo quindi un corrispondente file HTML (che chiameremo Main.htm) che verrà in esso caricato. A questo documento HTML andrà, dal lato dell'applicazione, associato un oggetto esposto allo scripting che sarà rappresentato dalla classe che chiameremo WebMain.

INTERPRETAZIONE QUERY

Per prima cosa la nostra classe esporrà un metodo che restituisce il risultato di una query al database. Questo processo si compone di due fasi distinte:
1. Estrazione dati in formato XML
2. Trasformazione con XSL in HTML

La funzione a ciò deputata è :

```
Public Function GetCustomers(ByVal filter As String) As
                                                String
'costruzione della query
    Dim sb As New System.Text.StringBuilder
    sb.AppendLine("select * from Customers")
'filter è il filtro gestito dal codice script
    If Not String.IsNullOrEmpty(filter) Then
'trasformazione del carattere jolly * nel corrispondente
                                                SQL %
        sb.AppendFormat("WHERE {0}",
                            filter.Replace("*", "%"))
        sb.AppendLine()
    End If
    sb.AppendLine("FOR XML AUTO") 'legge in XML
    Dim conn As New
```

```
SqlConnection(My.Settings.ConnectionString)
        Dim cmd As New SqlCommand(sb.ToString, conn)
'lettura dati
        Dim reader As SqlDataReader
        conn.Open()
        reader =
 cmd.ExecuteReader(CommandBehavior.CloseConnection)
        Dim queryResult As String = ""
        While reader.Read
            queryResult &= reader.GetString(0)
        End While
        reader.Close()
        queryResult = String.Format("<root>{0}</root>",
                                              queryResult)
        Dim strReader As New
                              IO.StringReader(queryResult)
'impostazione variabili di pagina
        Me.CurrentDocXml = New
                                 XPathDocument(strReader)
        Me.CurrentPage = 1
'chiamata funzione di trasformazione XSL
        Return TransformDoc()
    End Function
```

Come avrete notato il filtro viene affidato all'elaborazione da parte del codice script del documento HTML, quest'ultimo si presenterà pressappoco così:

```
function getSearchResult(){
        var filter = [];
// searchControls è l'array di textBox contenenti i termini
                                                 di ricerca
        for(var i=0;i<searchControls.length;i++){
                var filterValue =
                             searchControls[i].value;
                if(filterValue!="") {
//costruzione di un elemento della query basata sul
                             presupposto che il textBox
//abbia l'ID corrispondente al nome del campo
                             filter[filter.length] =
        searchControls[i].id + " LIKE '" + filterValue + "' " ;
                }
        }
//unione degli elementi della query
        currentFilter = filter.join(" AND ");
//chiamata alla funzione GetCustomers
        var searchResult =
              window.external.GetCustomers(currentFilter);
//utilizzo dell'HTML di ritorno per riempire una DIV
document.getElementById("result").innerHTML =
                                          searchResult;
}
```

NOTA

Per il funzionamento dell'esempio del codice allegato modificare opportunamente la stringa di connessione al database nelle impostazioni del progetto

La query XML produce un documento tipo:

```
<root>
  <Customers CustomerID="ALFKI"
                      CompanyName="Alfreds Futterkiste"
    ContactName="Maria Anders" ContactTitle="Sales
 Representative" Address="Obere Str. 57" City="Berlin"
PostalCode="12209" Country="Germany" Phone="030-
                        0074321" Fax="030-0076545"/>
...
</root>
```

dove quindi tutti i campi della tabella diventano attributi e ogni riga un elemento.
Questo risultato, come abbiamo visto nella funzione GetCustomers, viene processato da un'ulteriore funzione :

```
    Private _CurrentDocXsl As XPathDocument
    Private ReadOnly Property CurrentDocXsl() As
                                        XPathDocument
        Get
            If _CurrentDocXsl Is Nothing Then
                _CurrentDocXsl = New
                       XPathDocument(queryResult.xsl")
            End If
            Return _CurrentDocXsl
        End Get
    End Property
    Public Function TransformDoc() As String
        Dim transform As New XslCompiledTransform
        transform.Load(CurrentDocXsl)
        Dim args As New XsltArgumentList
        args.AddParam("currentPage", "", CurrentPage)
        Dim out As New System.Text.StringBuilder
        Dim strWriter As New IO.StringWriter(out)
        transform.Transform(CurrentDocXml, args,
                                             strWriter)
        Return out.ToString
    End Function
```

Questa funzione non fa altro che processare l'output XML della query con il foglio di stile XSL queryResult.xsl (anch'esso inserito nel progetto e definito come "da copiare" nella directory di output) di cui riportiamo un estratto:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output method="html" encoding="utf-
                                                 8"/>
        <xsl:param
                 name="currentPage">1</xsl:param>
        <xsl:param
                   name="pagesize">10</xsl:param>
        <xsl:param name="startPage"
            select="(($currentPage - 1) * $pagesize) +
                                   1"></xsl:param>
        <xsl:param name="totalPages"
            select="ceiling(count(/*/Customers) div
                               $pagesize)"></xsl:param>
        <xsl:template match="/">
```

```
<table cellpadding="2"
cellspacing="0" width="100%">
<thead>
<tr>
<th>Nome ditta</th>
<!--
altri campi -->
</tr>
</thead>
<tbody>
<xsl:for-each
select="/*/Customers[position()&gt;=$startPage and
position()&lt;($startPage + $pagesize)]">
<tr class="row1">
<td><xsl:value-of select="@CompanyName"/></td>
<!-- altri
campi -->
</tr>
</xsl:for-each>
</tbody>
<tfoot>
<!--
paginazione -->
</table>
</xsl:template>
</xsl:stylesheet>
```

ESPORTAZIONE IN EXCEL

Per l'esportazione in Excel potevamo utilizzare anche le librerie COM di excel, ma in questo caso nella macchina di destinazione avrebbe dovuto esserci Excel, abbiamo sfruttato quindi la possibilità che ha Excel (dalla versione 2000) di "capire" una tabella in formato HTML. Così la funzione nella classe WebMain:

```
Public Sub SaveAsExcel()
'dialogo per il nome del file
    Dim saveDialog As New SaveFileDialog
    saveDialog.InitialDirectory =
My.Computer.FileSystem.SpecialDirectories.MyDocuments
    saveDialog.DefaultExt = ".xsl"
    saveDialog.Filter = "File excel (*.xls)|*.xls"
    saveDialog.OverwritePrompt = True
    saveDialog.AddExtension = True
    If saveDialog.ShowDialog = DialogResult.OK Then
'scrittura nel file XLS dei dati HTML
My.Computer.FileSystem.WriteAllText(saveDialog.FileNam
e, Me.TransformDocExcel, False)
        If MsgBox("Vuoi vedere il risultato?",
    MsgBoxStyle.Question Or MsgBoxStyle.YesNo, "Export
Excel") = MsgBoxResult.Yes Then
'apertura a richiesta di EXCEL
System.Diagnostics.Process.Start(saveDialog.FileName)
        End If
    End If
End Sub
```

La funzione viene richiamata in modo diretto da HTML con :

```
<button
    onclick="window.external.SaveAsExcel()">Esporta in
excel</button>
```

La funzione TransformDocExcel presente nel codice di SaveAsExcel processa anch'essa l'output XML della query in un'unica tabella.

I RISULTATI

Risultato del nostro lavoro sarà una maschera iniziale di ricerca per quattro campi, vedi figura 9, con possibilità di utilizzare i caratteri jolly.

Fig. 9: ***Maschera di ricerca iniziale***

I risultati appariranno poi in forma di tabella (il risultato della trasformazione XSL), come potete vedere in figura 10

Tutti gli effetti (gradienti, roll-over ecc…) sono stati ottenuti con CSS, HTML e un po' di javascript.

Nome ditta	Contatto	Indirizzo	Città	Stato	Tel.	Fax
Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	Germany	030-0074321	030-0076545
Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	Mexico	(5) 555-4729	(5) 555-3745
Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	Mexico	(5) 555-3932	
Around the Horn	Thomas Hardy	120 Hanover Sq.	London	UK	(171) 555-7788	(171) 555-6750
Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	Sweden	0921-12 34 65	0921-12 34 67
Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	Germany	0621-08460	0621-08924
Blondesddsl père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	France	88.60.15.31	88.60.15.32
Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	Spain	(91) 555 22 82	(91) 555 91 99
Bon app'	Laurence Lebihan	12, rue des Bouchers	Marseille	France	91.24.45.40	91.24.45.41
Bottom-Dollar Markets	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	Canada	(604) 555-4729	(604) 555-3745

Fig. 10: ***interfaccia con i risultati***

CONCLUSIONI

Utilizzando il controllo WebBrowser con le possibilità che offre di "dialogo" con il documento HTML sottostante si è ottenuta un'interfaccia gradevole e funzionale in una frazione del tempo che ci sarebbe voluto ad implementare controlli personalizzati.

In più otteniamo l'ulteriore vantaggio che le modifiche all'interfaccia saranno implementabili semplicemente cambiando il codice HTML senza dover ricompilare il programma.

Francesco Smelzo

L'AUTORE

Francesco Smelzo è specializzato nello sviluppo in ambiente Windows con particolare riferimento ad applicazioni in ambiente .NET sia web-oriented che desktop.
Il suo sito web è www.smelzo.it. Come sempre è a disposizione per ricevere suggerimenti o richieste sull'articolo all'indirizzo di posta elettronica francesco@smelzo.it

APPUNTAMENTI: GESTIAMOLI IN JAVA

VI È MAI CAPITATO DI UTILIZZARE LA FUNZIONE CALENDARIO DI OUTLOOK? SAPEVATE CHE QUANDO INVITATE QUALCUNO AD UNA RIUNIONE UTILIZZATE UNO STANDARD? IN QUESTO ARTICOLO VEDREMO COME ACCEDERE AI CALENDARI

Molti di voi utilizzano Outlook. Qualcuno fra voi ne avrà provato la funzione calendario. Programmo una riunione e invio una mail a qualcuno. Nella mail in questione è contenuto un invito che mostra esplicitamente l'orario, il giorno, l'oggetto della riunione. L'utente che riceve la mail può accettare o meno l'invito. Molto probabilmente tutto questo potrebbe essere fatto manualmente. Ma poiché siamo nell'era dell'automazione, esiste un formato particolare con cui possono essere gestiti tutti gli eventi legati ad un calendario. Se avrete la pazienza di dare uno sguardo al contenuto testuale di una mail contenente un invito di Outlook scoprirete che al suo interno c'è qualcosa del genere:

```
BEGIN:VCALENDAR
PRODID:-//Microsoft Corporation//Outlook 11.0
                                    MIMEDIR//EN
VERSION:2.0
METHOD:REQUEST
BEGIN:VEVENT
ATTENDEE;ROLE=REQ-
            PARTICIPANT;RSVP=TRUE:MAILTO:OMISSIS
ATTENDEE;ROLE=REQ-
            PARTICIPANT;RSVP=TRUE:MAILTO: OMISSIS
ATTENDEE;ROLE=REQ-
            PARTICIPANT;RSVP=TRUE:MAILTO: OMISSIS
ATTENDEE;ROLE=REQ-
            PARTICIPANT;RSVP=TRUE:MAILTO: OMISSIS
ORGANIZER:MAILTO:Omissis
DTSTART:20061109T140000Z
DTEND:20061109T143000Z
LOCATION:Ufficio del capo
TRANSP:OPAQUE
SEQUENCE:0
UID:040000008200E00074C5B7101A82E0080000000
          00066AB1D5B02C70100000000000000000100
0000016E7B65839431F4E90F513D9BFDCB856
DTSTAMP:20061107T095436Z
DESCRIPTION:Data: giovedì 9 novembre 2006 15.00-
                              15.30 (GMT + 1.00 h)
Amsterdam\, Berlino\, Berna\, Roma\, Stoccolma\,
                                  Vienna.\nLuogo:
gianmarco\n\n*~*~*~*~*~*~*~*~*~*\n\n\n
SUMMARY:Programmazione budget annuale
PRIORITY:5
X-MICROSOFT-CDO-IMPORTANCE:1
CLASS:PUBLIC
BEGIN:VALARM
TRIGGER:-PT15M
ACTION:DISPLAY
DESCRIPTION:Reminder
END:VALARM
END:VEVENT
END:VCALENDAR
```

Quando Outlook riceve questa roba, la interpreta e la mostra all'utente in un formato comprensibile. In realtà il formato dei calendari è un formato standard, condiviso da una marea di applicazioni incluso Outlook. Questo stesso invito potrebbe essere letto anche da iCal di Mac, proprio perché il formato è uno standard. Nella realtà non è proprio tutto così semplice, perché qualcuno si ostina a voler modificare gli standard, ma in linea di massima il concetto appare chiaro. In questo articolo ci occuperemo appunto dello standard che sottende alla gestione dei calendari.

INTRODUZIONE A ICALENDAR

La definizione dello standard è contenuta nell'RFC 2445. Il formato è molto semplice. Non solo. È anche disponibile una libreria Java per manipolarlo, iCal4j (http://ical4j.sourceforge.net/). Questa offre sia un **parse** sia un modello a oggetti che permette di modificare dati iCalendar esistenti o di crearne di nuovi. È implementata anche la validazione in modo da assicurarsi che i dati contenuti siano consistenti con la specifica. Il *parser* è la componente software che si occupa di elaborare i dati iCalendar e di tradurli in una rappresentazione interna basata sugli oggetti. La

REQUISITI

Conoscenze richieste
Linguaggio Java

Software
Java2 SDK 1.4.2

Impegno

Tempo di realizzazione

struttura di questi oggetti invece è il *modello a oggetti*. Un semplice esempio di dati iCalendar è il seguente:

```
BEGIN:VCALENDAR
CALSCALE:GREGORIAN
X-WR-TIMEZONE;VALUE=TEXT:US/Pacific
METHOD:PUBLISH
PRODID:-//Apple Computer\, Inc//iCal 1.0//EN
X-WR-CALNAME;VALUE=TEXT:Example
VERSION:2.0
BEGIN:VEVENT
SEQUENCE:5
DTSTART;TZID=US/Pacific:20021028T140000
DTSTAMP:20021028T011706Z
SUMMARY:Coffee with Jason
UID:EC9439B1-FF65-11D6-9973-003065F99D04
DTEND;TZID=US/Pacific:20021028T150000
BEGIN:VALARM
TRIGGER;VALUE=DURATION:-P1D
ACTION:DISPLAY
DESCRIPTION:Event reminder
END:VALARM
END:VEVENT
END:VCALENDAR
```

Ciascuna linea di testo è chiamata *linea contenuto* e consiste in due elementi separati da un carattere di due punti (:).La prima parte è chiamata *proprietà* mentre il valore è la seconda parte. Per esempio, la riga VERSION:2.0 contiene la proprietà VERSION che vale 2.0. La proprietà può essere combinata con uno o più parametri, da cui è separata da punti e virgola (;). Ciascun parametro è separato da una virgola (,). Per esempio, la proprietà TRIGGER possiede il parametro VALUE che vale DURATION. Si noti che, per specifica, le linee terminano con il separatore CRLF.

CREARE UN CALENDARIO

Vediamo ora come utilizzare la libreria iCal4j per creare un nuovo calendario. I passaggi necessari non sono complessi e possono essere riassunti nei punti seguenti:

- creazione di un oggetto Calendar che rappresenta l'intero calendario. Questo verrà arricchito di eventi e di proprietà.
- Impostazione delle proprietà di base, come la versione del file e la tipologia di calendario.
- Aggiunta degli eventi e elementi to-do che rappresentano il contenuto di questo calendario ed eventuali configurazioni delle proprietà relative a ciascun evento.
- Produzione in output del file di calendario, su un qualsiasi stream di output, come per esempio un file su disco.

Vediamo ora come si traduce in codice questa sequenza di eventi realizzando un semplice programma che produce in output il file calendario.test con un solo evento, la cui descrizione è "Evento di prova" e impostato per l'istante corrente. Per inizializzare il calendario si fa come segue:

```
Calendar calendar = new Calendar();
calendar.getProperties().add(new ProdId("-//Ben
                                Fortuna//iCal4j 1.0//EN"));
calendar.getProperties().add(Version.VERSION_2_0);
calendar.getProperties().add(CalScale.GREGORIAN);
```

L'evento viene creato in questo modo, utilizzando l'oggetto **Date** definito nel **package net.fortuna.ical4j.model**. Questo viene inizializzato con la data corrente, utilizzando il costruttore di default dell'oggetto java.util.Date:

```
VEvent ev =
        new VEvent(new
          net.fortuna.ical4j.model.Date(new Date()),
        "Evento di prova");
```

Viene poi ottenuto l'elenco delle proprietà. Viene estratta la DTSTART che viene impostata al tipo Value.DATE. Questo è necessario per fare in modo che l'evento sia definito come "evento tutto il giorno":

```
PropertyList pl = ev.getProperties();
pl.getProperty(Property.DTSTART).getParameters().ad
                                        d(Value.DATE);
```

L'evento viene poi aggiunto al calendario con questa chiamata:

```
calendar.getComponents().add(ev);
```

La fase finale è quella che comporta la scrittura del calendario su un file. Per fare questo viene utilizzato un oggetto **FileOutputStream** per puntare al file fisico da creare. Per serializzare il calendario viene utilizzata la classe **CalendarOutputter**:

```
FileOutputStream fout = new
                FileOutputStream("calendario.test");
CalendarOutputter outputter = new
                        CalendarOutputter();
```

NOTA

Il Wiki di iCal4j (http://wiki.modularity.net.au/ical4j/) offre una serie di informazioni per iniziare a lavorare con la libreria, anche se non è ancora molto completo. Forse il lettore vorrà contribuire?

```
try {
        outputter.output(calendar, fout);
} catch (ValidationException e) {
        throw new IOException( e.getMessage() );
}
```

La classe CalendarOutputter dispone di due metodi **output**(). Il primo si aspetta un oggetto **OutputStream**, mentre il secondo un oggetto **Writer**. Questo permette di scrivere indifferentemente su un file binario (OutputStream) o di testo (Writer). Un altro aspetto interessante riguarda la possibilità di attivare la validazione dei dati che vengono scritti sul file. Questo si ottiene semplicemente passando il flag validating a true in fase di costruzione dell'oggetto.

ALTRE TIPOLOGIE DI EVENTI

Nell'esempio precedente abbiamo visto come creare un evento che dura tutto il giorno. In Figura 2 un evento di questo tipo è quello che riporta la descrizione "corso intensivo di nuoto" e viene rappresentato in modo diverso rispetto agli eventi che hanno una durata definita in termini di ore e minuti. Con iCalendar e iCal4j è possibile definire anche eventi di questo tipo. Nel codice che segue viene utilizzata la classe **Calendar** del **package java.util**. Questa è una classe della piattaforma Java che si occupa di gestire date e orari espressi all'interno di un calendario. **Calendar** è una classe astratta, ma è presente la classe concreta **GregorianCalendar** che implementa il calendario gregoriano. Il metodo **getInstance**() ritorna una istanza di calendario dove il giorno corrente "punta" al giorno odierno. Questa classe dispone poi di due metodi per cambiare il giorno corrente. Il metodo **add**() aggiunge una quantità a una delle numerose proprietà gestite dalla classe. Nel codice vengono aggiunti 7 giorni. Esiste una proprietà per ciascun elemento del datario (giorno, mese, anno, ora, minuti, secondi). In questo caso è stata utilizzata la costante DAY_OF_MONTH, che indica il giorno del mese. L'altro metodo offerto dalla classe Calendar per modificare la data corrente è **set**(). Questo imposta la proprietà a un valore arbitrario. Nell'esempio di codice viene impostata l'ora 9:30. Si noti la differenza tra **add**() e **set**(). Nel primo caso si aggiunge, anche valori negativi, al datario attuale. Nel secondo caso si imposta un valore arbitrario.
Il codice è il seguente:

```
java.util.Calendar cal =
                java.util.Calendar.getInstance();
//sposta il calendario alla settimana prossima
cal.add(java.util.Calendar.DAY_OF_MONTH, 7);
//appuntamento alle 9.30
cal.set(java.util.Calendar.HOUR_OF_DAY, 9);
cal.set(java.util.Calendar.MINUTE, 30);
VEvent meeting = new VEvent(cal.getTime(), 1000 *
                60 * 60, "Riunione di lavoro");
```

L'ora elaborata con la classe Calendar viene passata poi al costruttore dell'oggetto VEvent, a cui viene indicata anche la descrizione "Riunione di lavoro" e la durata dell'evento. In questo caso è un'ora (60 secondi * 60 minuti * 1000 millisecondi).

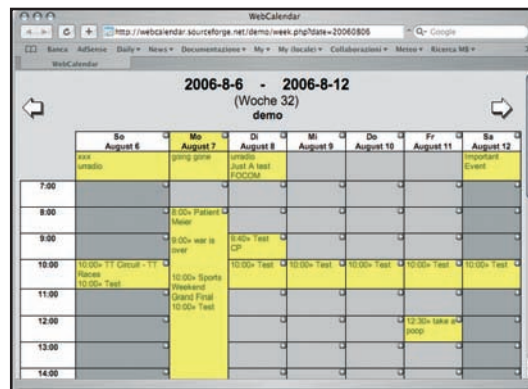



Fig. 1: *Un'applicazione d'esempio su sourceforcie*

NOTA

Per sperimentare con i calendari in formato iCalendar è possibile utilizzare un programma multipiattaforma e Open Source come Mozilla Sunbird (http://ftp.mozilla.org/pub/mozilla.org/calendar/sunbird/). È ancora in versione alfa, ma disponibile per Windows, Linux e Macintosh.

ALTRI ELEMENTI DI CALENDARIO

Fino a ora abbiamo visto come aggiungere eventi al calendario, siano essi di durata definita o relativi a tutto il giorno. In un calendario è anche possibile inserire allarmi, indicazioni di quando una persona è occupata o libera o elenchi di attività da svolgere. L'ultima di queste tipologie è di facile realizzazione, in quanto il codice da implementare è sostanzialmente uguale a quello di creazione di un evento. Cambia solo il nome della classe da utilizzare. Per creare l'attività "Riunione di lavoro" della durata di un'ora è necessario quindi utilizzare il seguente codice:

```
java.util.Calendar cal =
                java.util.Calendar.getInstance();
//sposta il calendario alla settimana prossima
cal.add(java.util.Calendar.DAY_OF_MONTH, 7);
//appuntamento alle 9.30
cal.set(java.util.Calendar.HOUR_OF_DAY, 9);
cal.set(java.util.Calendar.MINUTE, 30);
VToDo meeting = new VToDo(cal.getTime(), 1000 *
        60 * 60, "Riunione di lavoro");
```

Anche gli allarmi sono di facile creazione. Questa tipologia di eventi viene spesso utilizzata dal programma client per avvisare l'utente che c'è un'evento che inizia tra poco. Chi utilizza un qualche programma di calendario sarà abituato a questo

tipo di avvisi, spesso molto invadenti e insistenti! Per creare un allarme che deve essere presentato un'ora prima rispetto a un evento si può scrivere:

```
VAlarm reminder = new VAlarm(-1000 * 60 * 60);
```

Poi si può impostare la frequenza e il numero di avvisi. Nel codice seguente vengono impostate tre ripetizioni, una ogni 15 minuti. Questo vuol dire che il primo avviso verrà presentato un'ora prima, poi il successivo dopo un quarto d'ora e il terzo dopo ancora quindici minuti:

```
reminder.getProperties().add(new Repeat(3));
reminder.getProperties().add(new Duration(1000 * 60
                                               * 15));
```

Le proprietà dell'allarme possono anche memorizzare l'azione da intraprendere. Per esempio è possibile visualizzare un messaggio di avviso (DISPLAY). Le altre possibilità sono AUDIO, EMAIL e PROCEDURE. Queste sono definite, come il resto degli elementi presentati in precedenza, nel modello a oggetti di iCal4j che non fa altro che riportare nel mondo Java quanto espresso nelle specifiche di iCalendar:

```
reminder.getProperties().add(Action.DISPLAY);
reminder.getProperties().add(new
                Description("Riunione alle ore 9:30"))
```

Ovviamente la gestione degli allarmi è totalmente a carico dell'applicazione che elabora il file di dati iCalendar. È infatti compito suo, nel momento definito nell'allarme, operare l'intervento relativo all'azione indicata. Nel nostro esempio, l'azione è la visualizzazione del messaggio di avviso "Riunione alle ore 9:30". Si noti che l'allarme, utilizzando l'azione PROCEDURE, offre interessanti possibilità di sviluppo. iCal di Apple, per esempio, consente di impostare allarmi legati a eventi che aprano in automatico un documento, con un anticipo prefissato. Per esempio, si può impostare iCal in modo che quindici minuti prima dell'inizio dell'evento "Riunione di lavoro" aprano il file Word "agenda_riunione.doc" in modo che l'utente lo possa consultare o stampare. Un'altra possibilità è quella legata all'invio di messaggi di posta. Si ipotizzi di avere impostato un evento "Consegna articolo iCalendar per ioProgrammo". Sarebbe utile che un certo lasso di tempo prima della scadenza venga aperto un nuovo messaggio di posta pronto per l'invio del materiale. Questi sono solo alcuni esempi di come sia possibile sfruttare le potenzialità offerte da iCalendar.
L'ultimo elemento che analizziamo è rappresentato dalla classe VFreeBusy, che permette di definire i momenti in cui una persona è libera o occupata. Semplicemente, il costruttore di questa classe si aspetta due parametri: l'inizo e la fine del periodo "occupato". Per esempio, si ipotizzi che una persona sia in ferie da oggi per un mese. In questo caso si può scrivere come segue:

```
java.util.Calendar cal =
                        java.util.Calendar.getInstance();
Date inizio = cal.getTime();
cal.add(java.util.Calendar.WEEK_OF_YEAR, 4);
Date fine = cal.getTime();
VFreeBusy request = new VFreeBusy(inizio, fine);
```

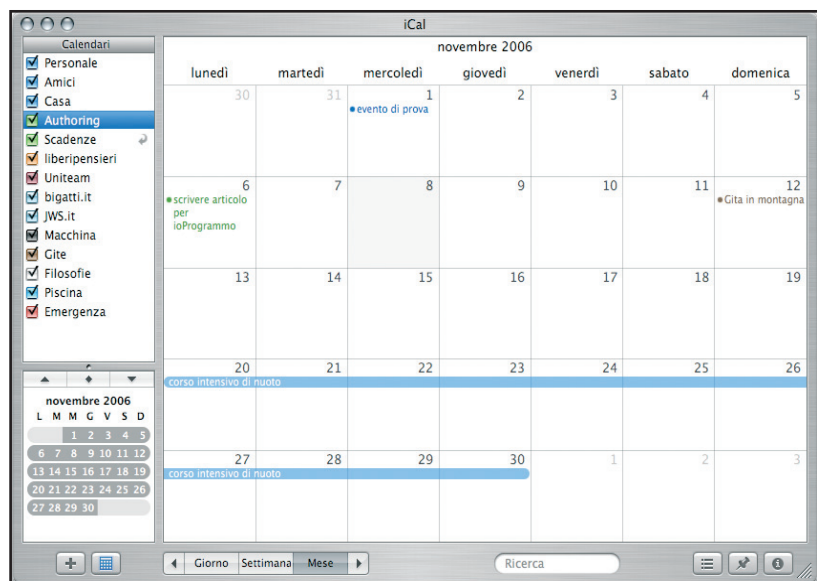



Fig. 2: ***Ical in esecuzione su Mac OS X***

CONCLUSIONI

Implementare il supporto allo standard iCalendar nelle proprie applicazioni Java è molto semplice grazie alla libreria iCal4j. Oltre che semplice, è anche molto utile, in quanto la vostra applicazione sarà in grado di comunicare con un ampio raggio di utenti, siano essi utilizzatori di Windows, Linux o Mac OS X e indipendentemente dal software utilizzato. Pensate al potenziale sviluppo della vostra applicazione gestionale. Potreste legare un evento a ogni progetto e pubblicare il calendario su un server web. Gli utenti che lo sottoscrivessero avrebbero sempre sotto controllo l'inizio e la fine prevista delle attività. Potreste presentare allarmi direttamente all'utente, utilizzando il loro programma di calendario. Un'altra possibilità è gestire l'elenco delle attività da svolgere. Quando l'utente le completa potrebbe indicarlo all'applicazione che si occuperebbe di rimuovere l'attività dall'elenco.

Massimiliano Bigatti

NOTA

Se siete programmatori a basso livello o avete voglia di conoscere di più le specifiche di iCalendar potete consultare direttamente la RFC2445, disponibile all'indirizzo http://www.ietf.org/rfc/rfc2445.txt.

FACCIAMO LEGGERE I PDF ALL'IPOD

IN QUESTO ARTICOLO CERCHEREMO DI UTILIZZARE JAVA E UNA FAMOSA LIBRERIA PER AUMENTARE LE POTENZIALITÀ DEL NOSTRO IPOD, TRASFORMANDO I FILE PDF IN DOCUMENTI LEGGIBILI SUL NOSTRO LETTORE MULTIMEDIALE

L'iPod è uno dei più famosi lettori mp3/mp4 dei nostri tempi, status symbol per molti, fedele compagno per altri. Su questo dispositivo è possibile ascoltare mp3, leggere note e vedere video. In rete è facile trovare programmi che consentono di convertire audio e video per questo dispositivo, ma un "geek" che si rispetti deve sempre fare qualcosa in più con il suo iPod. Perciò utilizzeremo il fedele Java e una manciata di librerie OpenSource per fare qualcosa che ci consenta di caratterizzare il nostro lettore come piace a noi…

L'APPLICAZIONE

Prima di tutto mettiamoci nell'ordine di idee che l'IPOD dispone di uno schermo da 2,5 pollici e una risoluzione di 320x240. Su questo schermo si possono visualizzare foto, video e testo. Il nostro scopo è riuscire a leggere sullo schermo dell'iPod dei file PDF. Poiché non abbiamo nessuna intenzione di scrivere da zero un visualizzatore di PDF, utilizzeremo un convertitore che ci consente di trasformare il formato PDF in qualcosa di visualizzabile sull'iPod. In particolare realizzeremo un programma che ci consente di:

- Convertire un PDF in testo semplice, che può facilmente essere letto come "nota"
- Trasformare un file PDF in una serie di immagini per poterlo leggere comodamente con il visualizzatore di immagini.

Per prima cosa, la nostra applicazione, che chiameremo IpodConverter, dovrà essere dotata di un'interfaccia grafica che mostri le diverse opzioni.

Conoscenze richieste
J2SE

Software
J2SE, JPedal

Impegno

Tempo di realizzazione

INTERFACCIA GRAFICA

L'utente deve poter scegliere tra 2 diverse opzioni. Iniziamo quindi scrivendo le porzioni di codice per un semplice Frame con 2 bottoni. L'applicazione è molto semplice, tuttavia vogliamo rimanere un po' nello stile IPOD, quindi utilizzeremo qualcosa per abbellire la nostra interfaccia. Inizialmente creeremo un semplice JFrame e aggiungeremo 2 JButton. A questi bottoni assoceremo degli **ActionListener**. Dovremo poi sviluppare le 2 funzionalità associate ai diversi bottoni

```
jButton1.setText("PDF to image");
jButton1.addActionListener(new
                    java.awt.event.ActionListener() {
    public void
  actionPerformed(java.awt.event.ActionEvent evt) {
        action1(evt);
    }
});
jButton2.setText("PDF to text");
jButton2.addActionListener(new
                    java.awt.event.ActionListener() {
    public void
  actionPerformed(java.awt.event.ActionEvent evt) {
        action2(evt);
    }
});
jButton3.setText("Exit");
jButton3.addActionListener(new
                    java.awt.event.ActionListener() {
    public void
  actionPerformed(java.awt.event.ActionEvent evt) {
        action3(evt);
    }
});
```

Abbiamo aggiunto anche un terzo bottone per uscire dall'applicazione. Di seguito mostriamo lo scheletro dei tre metodi che verranno richiamati quando uno dei bottoni verrà premuto.

```
private void action3(java.awt.event.ActionEvent evt) {
        System.out.println("Uscita dall'applicazione");
        System.exit(1);
}
private void action2(java.awt.event.ActionEvent evt) {
```

```
        // Codice per l'estrazione del testo dai PDF
}
private void action1(java.awt.event.ActionEvent evt) {
        // Codice per la trasformazione da PDF a
                                        immagini
}
```

Chiaramente l'interfaccia grafica è molto scarna e potremmo abbellirla con qualche Look&Feel più accattivamente. I Look&Feel Swing permettono di cambiare il layout dell'interfaccia grafica in maniera semplice. Per questa applicazione è stato scelto di utilizzare il L&F substance, che potete trovare disponibile per il download all'indirizzo https://substance.dev.java.net . Il layout di default di questo L&F assomiglia molto alle gui che possiamo vedere su MacOS X. Per utilizzarlo nella nostra interfaccia non dobbiamo far altro richiamare la classe UIManager e dichiarare di volerlo utilizzare.

```
    try
    {
UIManager.setLookAndFeel("org.jvnet.substance.Substan
                                    ceLookAndFeel");
    }
    catch(Exception e) {
        System.out.println(e.toString());
    }
```

Potete ammirare quindi la nostra interfaccia nell'immagine di seguito.

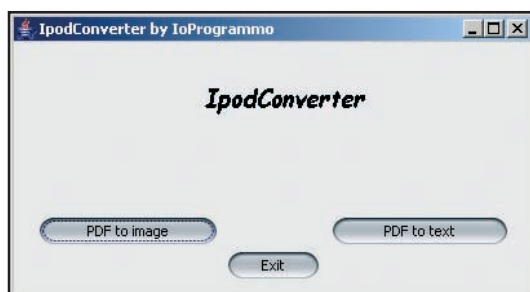



Fig. 1: **Screenshot della nostra applicazione**

Torniamo ora a concentrare la nostra attenzione sulle funzionalità del nostro programma, implementando una dopo l'altra le funzionalità dichiarate.

DA PDF A TESTO

Il pdf è un formato molto comune ed esistono al giorno d'oggi molti tool che permettono di salvare un documento in questo formato. Purtroppo questo tipo di file non è supportato dall'IPOD per la visualizzazione, quindi dobbiamo ingegnarci per ottenere come prima cosa il testo di un intero file pdf e poterlo leggere nelle Note dell'IPOD. Prima di tutto dobbiamo utilizzare una libreria esterna che ci permetta di interagire con file pdf, visto che le librerie standard di Java non prevedono niente al riguardo. Nel panorama delle librerie opensource Java disponiamo di diverse scelte. In questo articolo utilizzeremo JPedal, una libreria opensource rilasciata con licenza duale: GPL e commerciale. Questa libreria nasce nel lontano 1997 e nel tempo è stata costantemente aggiornata, sinonimo di affidabilità del gruppo che ne sostiene lo sviluppo. Attualmente è arrivata alla versione 2.8. Possiamo scaricarla dal sito ufficiale http://www.jpedal.org/. Alcune fra le feature disponibili con Jpedal sono le seguenti:

- Supporto alle form pdf (form che possiamo trovare all'interno di file pdf e ci permettono di inserire valori)
- Estrazione del testo
- Estrazione delle immagini
- Visualizzatore pdf
- Support per il Font
- Support per il Mac
- API alternativa per la stampa

Fig. 2: **Homepage del sito di JPedal**

Iniziamo quindi a sviluppare la parte del nostro programma che permetterà di estrarre il testo dai file pdf. Il metodo relativo all'estrazione del testo è **action2**() che andiamo qui sotto ad implementare

```
JFileChooser fileChooser = new JFileChooser(new
                                    File("C:\\"));
fileChooser.addChoosableFileFilter(new FiltroPdf());
// Open file dialog.
fileChooser.showOpenDialog(this);
File f=fileChooser.getSelectedFile();
if (f!=null) {
        System.out.println("Inizia estrazione testo");
        EstrattoreTesto et=new
            EstrattoreTesto(f.getAbsolutePath(),this);
        et.estrai();
}
```

Prima di tutto facciamo in modo che l'utente possa

Una libreria offerta dalla Apple, QuickTime for Java, può essere usata per convertire file audio-video, adattandoli allo standard riconosciuto dal IPOD.
http://developer.apple.com/quicktime/qtjava/
http://www.onjava.com/pub/a/onjava/2003/05/14/qtj_reintro.html
http://www-128.ibm.com/developerworks/wireless/library/wi-mvideo/?ca=dgr-lnxw17CreateIPODvideo

scegliere il file PDF da elaborare. Lo facciamo utilizzando la classe Java Swing JFileChooser e applicandovi un filtro che restringa la scelta a directory e file con estensione .pdf

```
package com.ioprogrammo.ipodconverter;
import java.io.*;
public class FiltroPdf extends
                    javax.swing.filechooser.FileFilter {

  public boolean accept(File file) {
    String filename = file.getName();
    return
       (filename.endsWith(".pdf")||(file.isDirectory()));
  }
  public String getDescription() {
    return "*.pdf";
  }

}
```

Una volta che l'utente ha selezionato un file pdf, siamo pronti per estrarne il testo. Quasi sicuramente si tratterà di un processo lungo, quindi la classe che se ne occuperà dovrà essere un Thread. In questo modo non ci saranno blocchi dell'interfaccia grafica. All'interno della libreria JPedal è presente un esempio utile (ExtractTextAsWordlist) che abbiamo riadattato per una migliore integrazione con il nostro programma grafico.

```
package com.ioprogrammo.ipodconverter;

import java.io.*;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;
import javax.swing.JOptionPane;
import org.jpedal.PdfDecoder;
import org.jpedal.exception.PdfException;
import org.jpedal.exception.PdfSecurityException;
import org.jpedal.grouping.PdfGroupingAlgorithms;
import org.jpedal.objects.PdfPageData;
import org.jpedal.utils.LogWriter;
import org.jpedal.utils.Strip;
public class EstrattoreTesto extends Thread {
      private String filename;
      private int wordsExtracted;
      private PdfDecoder decodePdf;
      private String outputDir;
      private IpodConverter frame;

      public EstrattoreTesto(String
                filename,IpodConverter frame) {
          this.filename=filename;
          this.frame=frame;
      }

      public void estrai() {
       this.start();
      }
```

Nel metodo run() di questo Thread andremo ad estrarre il testo. La prima cosa che facciamo è aprire il file utilizzando la classe *PdfDecoder* della libreria JPedal

```
public void run() {
      PdfDecoder.useTextExtraction();
      outputDir ="c:\\IpodConverter\\";
      try {
        decodePdf = new PdfDecoder(false);
        decodePdf.setExtractionMode(1);
        decodePdf.init(true);
        PdfGroupingAlgorithms.useUnrotatedCoords
                                        = false;
        System.out.println("Apertura file :" +
                                     filename);
        decodePdf.openPdfFile(filename);
      }
      catch(Exception ex) {
        System.out.println(ex.toString());
      }
```

Una volta aperto il file dobbiamo vedere se sono presenti delle restrizioni sul file pdf, tipo password o permessi negati

```
if(decodePdf.isEncrypted() &&
                !decodePdf.isPasswordSupplied() &&
                 !decodePdf.isExtractionAllowed()) {
            System.out.println("Impossibile eseguire
        l'estrazione per politiche di sicurezza sul file");
 }
```

Se il pdf non ricade in nessuno di questi casi possiamo iniziare a prendere il testo. Questo lavoro viene svolto pagina per pagina, dando le coordinate della pagina ad una classe di JPedal che estrae un vettore con il testo. Quello che noi dobbiamo fare è calcolare quante pagine sono presenti, fare un ciclo for e su ogni pagina ottenere il testo e scriverlo su file.

```
else {
        boolean flag = true;
        int j = decodePdf.getPageCount();
        try {
          for(int k = ((flag) ? 1 : 0); k < j + 1;
                                               k++)
          {
            decodePdf.decodePage(k);
            PdfGroupingAlgorithms
pdfgroupingalgorithms = decodePdf.getGroupingObject();
            PdfPageData pdfpagedata =
                        decodePdf.getPdfPageData();
            int l = pdfpagedata.getMediaBoxX(k);
```

NOTA

E' possibile vedere altri Look&Feel, per migliorare la grafica della nostra applicazione, collegandosi al sito http://www.javootoo.com

```
int i1 =
    pdfpagedata.getMediaBoxWidth(k) + l;
int j1 = pdfpagedata.getMediaBoxX(k);
int k1 =
    pdfpagedata.getMediaBoxHeight(k) - j1;
System.out.println("Estrazione pagina
    "+k);
Vector vector = null;
try {
    vector =
pdfgroupingalgorithms.extractTextAsWordlist(l, k1, i1, j1,
    k, false, true, "&:=()!;.,\\/\"\"""");
}
catch(PdfException ex)
{
    decodePdf.closePdfFile();
    System.err.println("Eccezione = " +
    ex );
}
if(vector == null) {
    System.out.println("Testo non
    trovato");
}
else {
    File file = new File(outputDir);
    if(!file.exists())
        file.mkdirs();
    int l1 = vector.size() / 5;
    wordsExtracted = wordsExtracted +
    l1;
    OutputStreamWriter
  outputstreamwriter = new OutputStreamWriter(new
FileOutputStream(outputDir + "pagina-" + k + ".txt"),
    "UTF-8");
    String s2;
    int i2;
    int j2;
    int k2;
    int l2;
    for(Iterator iterator =
     vector.iterator(); iterator.hasNext();
     outputstreamwriter.write(s2+" ")) {
        s2 = (String)iterator.next();
        s2 = Strip.convertToText(s2);
        i2 =
    (int)Float.parseFloat((String)iterator.next());
        j2 =
    (int)Float.parseFloat((String)iterator.next());
        k2 =
    (int)Float.parseFloat((String)iterator.next());
        l2 =
    (int)Float.parseFloat((String)iterator.next());
    }
    outputstreamwriter.close();
}
decodePdf.flushObjectValues(false);
}
```

```
    }
    catch(Exception ex)
    {
        decodePdf.closePdfFile();
        System.err.println("Eccezione: " + ex);
        ex.printStackTrace();
    }
    decodePdf.flushObjectValues(true);
    System.out.println("Testo letto");
  }
  decodePdf.closePdfFile();
  decodePdf = null;
JOptionPane.showMessageDialog(frame,"Estrazione
    effettuata");
}
```

Il salvataggio delle diverse pagine viene effettuato nella cartella C:\\IpodConverter, salvando ogni pagina su un file diverso. Quando la conversione termina viene utilizzata la classe JOptionPane per comunicare la fine del lavoro. Questi file di testo che abbiamo realizzato possono essere letti facilmente su un IPOD come note, trasferendoli nella cartella Notes che l'IPOD mette a disposizione quando lo esploriamo come un disco esterno. Chiaramente la conversione non è perfetta perché non tutti i pdf sono uguali. In molte prove la conversione è andata bene, in alcuni casi vengono mischiati nel testo dei caratteri strani che JPedal non riesce ad eliminare.

DA PDF A IMMAGINI

La maggior parte dei file pdf che ci capita di leggere sono alla fine delle presentazioni. Proprio per questo motivo sarebbe molto interessante poter trasferire questi file sul nostro IPOD per poterli visualizzare quando vogliamo. Possiamo immaginare il file pdf come una serie di slide, una per pagina, che possiamo riprodurre sequenzialmente. Anche in questo caso la libreria JPedal può essere d'aiuto. Infatti utilizzando questa libreria è possibile vedere ogni pagina di un pdf come un'immagine e quindi salvarla e ridimensionarla come vogliamo. Anche in questo caso deleghiamo tutto il lavoro ad un classe esterna che agisce come Thread per non bloccare l'interfaccia grafica.

```
JFileChooser fileChooser = new JFileChooser(new
    File("C:\\"));
fileChooser.addChoosableFileFilter(new FiltroPdf());
fileChooser.showOpenDialog(this);
File f=fileChooser.getSelectedFile();
if (f!=null) {
    System.out.println("Inizia estrazione
    immagini");
    EstrattoreImmagini ei=new
        EstrattoreImmagini(f.getAbsolutePath(),this);
```

NOTA

Esistono già altri programmi e metodi che permettono di convertire diversi formati di file per l'IPOD

http://www.nullriver.com/index/products/moviepod
http://www.alloksoft.com/ipodconverter.htm
http://www.engadget.com/2004/11/16/how-to-put-powerpoint-on-your-ipod-photo/

```
        ei.estrai();
}
```

La classe **EstrattoreImmagini** viene inizializzata come **EstrattoreTesto**, passando il riferimento alla classe **IpodConverter**. Come prima viene istanziato un **PdfDecoder**, ma in questo caso viene specificato anche un parametro addizionale riguardamento la modalità di estrazione

```
try {
        decodePdf = new PdfDecoder(true);
        decodePdf.setExtractionMode(0, 72,dpi/72);
        decodePdf.openPdfFile(filename);
} catch (Exception e) {
        System.out.println("Errore nell'inizializzazion di
                        PdfDecoder: "+e.toString());
}
```

A questo punto, se non ci sono problemi di sicurezza del pdf (criptato o con password) possiamo iniziare a scorrere tutte le pagine in un ciclo for e prendere l'immagine relativa ad ogni pagina

```
File output = new File("c:\\IpodConverter\\");
        if (output.exists() == false)
          output.mkdirs();

        int end = decodePdf.getPageCount();
        try {
          for (int page = 1;page < end + 1;page++) {
            String image_name ="page_" + page;
            PdfFileInformation
 currentFileInformation=decodePdf.getFileInformationData
                                        ();

            BufferedImage image_to_save
                  =decodePdf.getPageAsImage(page);
```

A questo punto dovremmo effettuare il ridimensionamento dell'immagine. Dal punto di vista della qualità, trasformando l'immagine perdiamo qualcosa. Inoltre difficilmente otterremo un'immagine della lunghezza e larghezza giusta per l'IPOD. Infine i file delle immagini potrebbero essere utili anche in altri contesti, diversi dall'IPOD. Insomma diverse cose fanno pensare che sia inutile scrivere questo pezzo di codice nel nostro programma, anche perché iTunes ottimizza le immagini quando le vogliamo trasferire sul dispositivo. Comunque giusto per curiosità quello che avremmo dovuto fare viene riportato nel seguente codice

```
int
 newWidth=image_to_save.getWidth()*ridimensionament
                                      o/100;
Image scaledImage=
        image_to_save.getScaledInstance(newWidth,-
                1,BufferedImage.SCALE_SMOOTH);
image_to_save = new
 BufferedImage(scaledImage.getWidth(null),scaledImage.
        getHeight(null) , BufferedImage.TYPE_INT_RGB);
Graphics2D g2 = image_to_save.createGraphics();
g2.drawImage(scaledImage, 0, 0,null);
```

La variabile di ridimensionamento ci avrebbe permesso di stabilire la percentuale di riduzione dell'immagine. Passiamo ora al salvataggio dell'immagine sul nostro computer per concludere la funzionalità

```
boolean
   failed=decodePdf.getObjectStore().saveStoredImage(
                        "c:\\IpodConverter\\" + page
                                    + image_name,
                        image_to_save,
                        true,
                        false,
                        format);
```

Per poter utilizzare queste classi grafiche dobbiamo inserire nel nostro classpath anche JAI (Java Advanced Imaging,), la libreria della SUN che aggiunge funzionalità grafiche a quelle standard (http://java.sun.com/products/java-media/jai) .

CONCLUSIONI

Facendo un test della nostra applicazione vediamo quanto sia semplice ora poter leggere pdf in diverse modalità sul nostro IPOD.

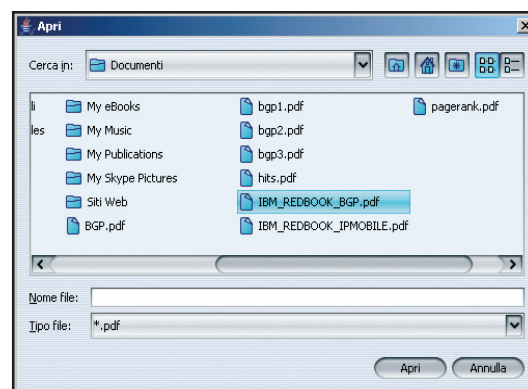



Fig. 3: ***Selezione del pdf da convertire***

Chiaramente questa è un'applicazione che può essere migliorata molto, inserendo il supporto ad altri formati di file, ad esempio la conversione di video in mp4, creazione di immagini a partire da altri file oltre ai pdf (powerpoint ad esempio) etc. etc. Come sempre la creatività è il punto di partenza per le applicazioni più interessanti.

Federico Paparoni

L'AUTORE

L'autore, Federico Paparoni, può essere contattato per suggerimenti o delucidazioni all'indirizzo email federico.paparoni@javastaff.com

PROGRAMMARE .NET CON PYTHON

LA DIFFUSIONE DI PYTHON COME LINGUAGGIO DI PROGRAMMAZIONE HA QUASI RAGGIUNTO QUELLA DI .NET, E CON IRONPYTHON È ORA POSSIBILE SCRIVERE SCRIPT IN PYTHON E COMPILARLI PER LA PIATTAFORMA MICROSOFT

Python è un linguaggio di programmazione potente, di facile apprendimento, e che permette di utilizzare diversi paradigmi di sviluppo. In particolare, il suo approccio alla programmazione orientata agli oggetti è semplice ma estremamente efficace. Grazie alla sua sintassi, alla tipizzazione dinamica, ed alla natura di linguaggio interpretato, si presta particolarmente per lo scripting e lo sviluppo rapido di applicazioni multipiattaforma.
Con IronPython, il linguaggio Python incontra il mondo .NET. IronPython è infatti un'implementazione del linguaggio per il CLR, integrato perfettamente con il mondo .NET, rendendo dunque le librerie del framework Microsoft utilizzabili da Python, mantenendo allo stesso tempo la totale compatibilità con il linguaggio python originale.
In questo articolo si vedrà come utilizzare la console interattiva di (Iron)Python, come utilizzare gli assembly .NET standard in Python, e come scrivere invece librerie in Python per integrarle e utilizzarle nelle applicazioni .NET.

LA CONSOLE DI PYTHON

Una volta scaricato il pacchetto dei binari di IronPython l'installazione non richiede nessuna operazione particolarmente complessa. Basta decomprimere i file in una cartella a vostra scelta, ed avrete già tutti gli strumenti necessari a disposizione.
Python possiede una console interattiva, una caratteristica fondamentale del linguaggio che dunque non poteva mancare ad IronPython. Fra i file sarà dunque presente l'eseguibile ipy.exe, che una volta lanciato eseguirà proprio l'interprete, presentandovi un prompt dei comandi pronto a ricevere ed eseguire istruzioni in python, come mostrato di seguito:

```
IronPython 1.0 (1.0.61005.1977) on .NET
                                          2.0.50727.42
Copyright (c) Microsoft Corporation. All rights
                                              reserved.
>>>
```

Chi conoscesse Python, la sua sintassi e le sue librerie può anche saltare questa parte, chi invece non ha mai scritto una sola riga di Python, e non conosce dunque la console interattiva, può eseguire i seguenti esempi almeno per provarne il funzionamento.
La console interattiva può essere utilizzata come calcolatrice! Si provi ad esempio ad inserire delle istruzioni aritmetiche e :

```
>>> 1+1
2
>>> 343*2
686
>>>
```

Al prompt possiamo anche eseguire del codice più articolato, che verrà immediatamente interpretato. Per esempio un ciclo *for* in python per stampare i primi dieci numeri viene scritto così:

```
>>> for i in range(1,10):
...         print i
```

Si noti come Python non utilizza alcun delimitatore di blocco, per esempio le parentesi graffe di C#, ma è basato esclusivamente sulla corretta indentazione del codice. Quindi si presti attenzione alla scrittura del codice. Per utilizzare le librerie di python dette moduli, si utilizza la funzione import, per esempio così:

```
>>> import sys
```

Per conoscere il contenuto del modulo sys si può utilizzare la funzione dir:

```
>>> dir(sys)
```

INTERPRETE INTERATTIVO

Per potere eseguire l'interprete interattivo da qualunque locazione del file system, è suggerito aggiungere il percorso contenente i file di IronPython, e quindi ipy.exe, alla variabile PATH di sistema. L'eseguibile ipiw.exe serve a lanciare l'interprete senza un prompt dei comandi, quindi si utilizza per applicazioni Windows.

Il modulo, fra gli altri, conterrà degli attributi, per esempio version che rappresenta la versione dell'ambiente. Per visualizzare il valore basta eseguire il comando seguente:

```
>>> sys.version
'IronPython 1.0 (1.0.61005.1977) on .NET
                                  2.0.50727.42'
>>>
```

UTILIZZARE LE LIBRERIE .NET

Per utilizzare le classi librerie del framework .NET dalla console interattiva di Python, o da uno script Python qualunque, basta utilizzare ancora la funzione **Import**.
Per esempio, se si vuol utilizzare la classe Console del namespace System, si può scrivere:

```
>>> import System
>>> System.Console.WriteLine('Hello')
Hello
>>>
```

In questo caso è necessario utilizzare il nome completo, cioè comprensivo di System. In caso contrario si otterrebbe infatti un errore di mancata definizione:

```
>>> Console.WriteLine('Hello')
Traceback (most recent call last):
  File , line 0, in <stdin>##31
NameError: name 'Console' not defined
```

La funzione import può essere però utilizzata anche per importare il suo contenuto oppure una particolare classe nel namespace globale. Per esempio:

```
>>> from System import *
>>> Console.WriteLine('Hello')
Hello
>>>
```

Per eseguire uno script python, basta salvarlo in un file con estensione .py e poi eseguire il comando

```
ipy nomefile.py
```

Proviamo a scrivere un semplice esempio che utilizza le librerie .NET, ed in particolare i generics.
Per questi ultimi è necessario utilizzare le parentesi quadre per il tipo argomento, e non < e > come si fa in .NET.

```
#esempio script ironpython
from System import *
from System.Collections.Generic import *

lista=List[String]()
lista.Add('ciao')
lista.Add('da')
lista.Add('IoProgrammo')

for elemento in lista:
        Console.Write(elemento+' ')
```

Salvate il codice precedente nel file esempio1.py ed eseguite 'ipy esempio1.py'. Il risultato sarà la visualizzazione della stringa 'Ciao da IoProgrammo'.
IronPython può importare direttamente solo alcune delle librerie di .NET, cioè quelle standard. Per le rimanenti, oppure per quelle create da noi in C#, in VB.NET o in un qualunque linguaggio .NET, è necessario qualche operazione in più.
Innanzitutto creiamo un semplice assembly .NET, per esempio in C#. Create un file con la seguente classe C#, il cui unico metodo calcola il quadrato di un intero:

```
using System;

public class MathLib
{
        public int Quadrato(int i)
        {
                return i*i;
        }
}
```

Dopo aver salvato tale classe in un file, per esempio MathLib.cs, bisogna compilare il file in modo da ottenere una libreria. Con il compilatore da riga di comando csc basta eseguire il comando:

```
csc /target:library MathLib.cs
```

Il risultato sarà l'assembly MathLib.dll. Adesso creiamo un altro script python all'interno del quale utilizzeremo l'assembly MathLib, e per far ciò bisogna utilizzare il modulo clr, ed in particolare il metodo **AddReferenceToFile**:

```
import clr
clr.AddReferenceToFile("MathLib.dll")
import MathLib
m=MathLib()
for i in range(1,11):
        print m.Quadrato(i)
```

Salvando lo script nel file esempio2.py e dandolo

in pasto a ipy.exe si otterrà in output il quadrato dei primi dieci numeri interi.

UTILIZZO DELLE WINDOWS FORMS

Ora che siamo entrati nei meccanismi fondamentali di IronPython e abbiamo scoperto come utilizzare le librerie .NET, possiamo anche affrontare la programmazione Windows Forms.
Non è niente di particolare, conoscendo le basi. Il seguente script crea una classica Form e la utilizza come finestra principale dell'applicazione, si noti l'utilizzo del metodo AddReferenceBy PartialName del modulo clr utilizzato per importare l'assembly **System.Windows.Forms** dalla GAC:

```
import clr
clr.AddReferenceByPartialName("System.Windows.Forms"
                                                    )
from System.Windows.Forms import *
class MainForm(Form):
        def __init__(self):
        self.Text = 'Hello World'
Application.Run(MainForm())
```

Nell'esempio viene creata una classe MainForm derivata dalla classe base **Form**. Il metodo **__init__** è il costruttore della classe, all'interno del quale viene solo inizializzato il testo sulla barra del titolo.
Si noti che ogni metodo di classe ha come primo parametro la variabile **self**, che rappresenta l'istanza della classe attuale (il this di C#).
Questo semplice esempio potrebbe già entusiasmare gli amanti di Python, invogliandoli a provare anche lo sviluppo per la piattaforma .NET, senza abbandonare il proprio linguaggio preferito, come d'altronde fa chi sviluppa in C#, o in VB.NET, o in C++, e così via.
Tra l'altro ciò consente di creare applicazioni grafiche senza dover far ricorso ad altre librerie, in quanto i namespace System.Windows.Forms, System.Drawing e così via hanno tutto ciò che serve. Proviamo dunque un'applicazione più complessa, con gestione degli eventi e disegno su una Form, riprendendo l'esempio precedente.

```
class MainForm(Form):
def __init__(self):
                self.Text = 'Hello World'
                self.FormClosing += self.formClosing
                self.InitializeComponents()
```

nel costruttore viene gestito l'evento Form Closing della Form ed invocato il metodo **InitializeComponents**, che crea l'interfaccia grafica come qui di seguito:

```
def InitializeComponents(self):
        bt=Button()
        bt.Text="Clicca qui"
        bt.Click += self.bt_Click
        bt.Location=Point(10,10)
        bt.Size=Size(80,30)
        self.Controls.Add(bt)
        self.MouseDown+=self.formMouseDown
```

Nel metodo InitializeComponents viene creato un pulsante **bt**, impostate le sue proprietà, ed il metodo che gestisce l'evento Click, definito fra poco. Poi il pulsante viene aggiunto alla collezione **Controls** della Form. L'ultima istruzione invece aggiunge il metodo **formMouseDown** come gestore dell'evento **MouseDown** della Form.

```
def formClosing(self, sender, e):
                result = MessageBox.Show
        ("Vuoi uscire?", "", MessageBoxButtons.YesNo)
                if result == DialogResult.No:
                        e.Cancel = True
                return
```

Il metodo **formClosing** gestisce l'evento **FormClosing**, che viene scatenato quando l'utente tenta di chiudere la finestra. In questo caso viene visualizzata una MessageBox con i pulsanti sì e no, per confermare o meno l'uscita dall'applicazione. In caso negativo viene impostato il valore **e.Cancel** a true per annullare l'evento di chiusura.

```
def bt_Click(self,sender,e):
        MessageBox.Show("Hai cliccato il pulsante!")
```

Il metodo **bt_Click** gestisce come visto l'evento

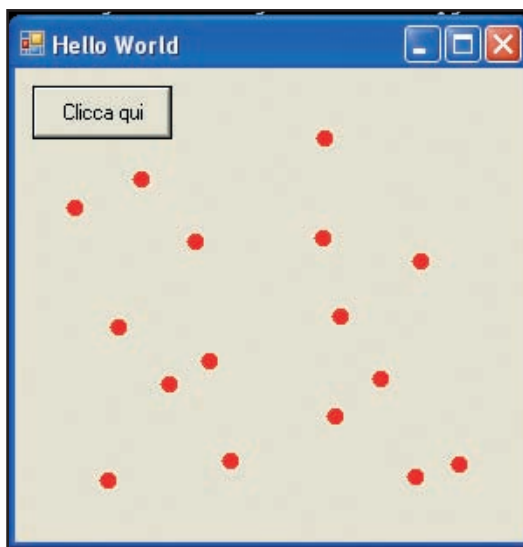



Fig. 1: un'applicazione Windows Forms in python

NOTA

IRONPYTHON SU CODEPLEX

Il progetto IronPython, è ospitato su CodePlex, dal quale potete ottenere sia il pacchetto dei binari, ma anche i sorgenti del progetto, insiemi a diversi esempi e tutorial. L'indirizzo ufficiale del progetto IronPython è http://www.codeplex.com/Wiki/View.aspx?ProjectName=IronPytho.

di clic sul pulsante **bt**, non fa niente di particolare in questo caso, solo visualizzare una MessageBox per verificare che in effetti l'evento viene gestito.
Infine vediamo il metodo formMouseDown, che fa uso delle funzioni di disegno del namespace System.Drawing.

```
def formMouseDown(self,sender,e):
        gfx=self.CreateGraphics()
        gfx.FillEllipse(Pens.Red,e.X,e.Y,10,10)
```

Ad ogni clic sulla superficie della Form viene disegnato un cerchio rosso. Dopo aver salvato il codice precedente in un file winapp.py, basta lanciare il comando ipy.exe winapp.py. La figura seguente mostra l'applicazione in esecuzione.

IL MOTORE DI PYTHON

È possibile utilizzare l'engine di Python all'interno di una classica applicazione .NET, ed utilizzare quindi le sue funzionalità. Abbiamo per esempio visto come utilizzare la console di python come calcolatrice, e ciò suggerisce che sarebbe possibile utilizzare l'interprete per valutare delle espressioni oppure per dotare le applicazioni .NET di funzioni di scripting, cosa per la quale python sta probabilmente divenendo il linguaggio preferito dai più.
Valutare un'espressione è un'operazione che suggerisce come possibile applicazione il disegno del grafico di una funzione. Come esempio quindi, si realizzerà una semplice applicazione

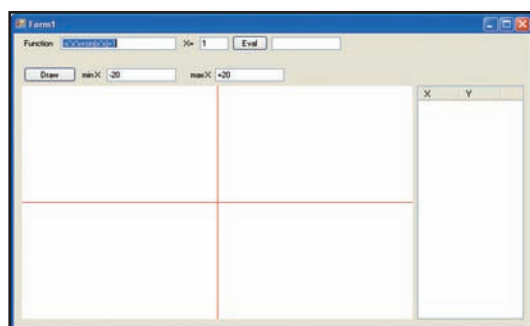



Fig. 2: L'interfaccia utente per lo studio di una funzione

Windows, che permetta di disegnare tale grafico, nel caso di una funzione ad una sola variabile x, sfruttando python per calcolarne i valori su un dato dominio di x.
La classe che rappresenta e permette di utilizzare il motore dell'interprete Python è PythonEngine, le cui istanze possono essere controllate da una qualunque applicazione host scritta in .NET.
Un'istanza PythonEngine permette di effettuare qualunque operazione eseguibile da Python, e dunque ognuna di quelle che abbiamo visto utilizzando la console interattiva.
Nel nostro caso, un PythonEngine servirà ad eseguire una semplice istruzione di assegnamento alla variabile x, lungo tutto il suo dominio e poi a valutare un'espressione rappresentata da una funzione ad una variabile.
Dopo aver preparato un'interfaccia grafica in C#, che permetta di inserire la funzione e l'intervallo dei domini, e che contenga un pannello sul quale disegnare la funzione (rappresentata in figura 2, codice presente nel CD allegato), andiamo a vedere come utilizzare la classe PythonEngine.
Creiamo una classe helper che crei e mantenga l'istanza PythonEngine, in questo modo:

```
class IronPythonHelper
{
    private PythonEngine engine;
    public IronPythonHelper()
    {
        InitializePyEngine();
    }
    void InitializePyEngine()
    {
        engine = new PythonEngine();
//esegui altre inizializzazioni
        engine.Import("sys");
        engine.Execute("from math import *");
    }
}
```

Nel metodo InitializePyEngine sarà possibile eseguire operazioni di inizializzazione addizionali, come per esempio importare moduli, in questo caso l'importazione del modulo math è praticamente obbligatorio per calcolare funzioni più complesse, come quelle trigonometriche.
Il processo sarà quello di calcolare il valore di una funzione, su tutti i punti del dominio della variabile X. Quindi prima di tutto si assegnerà alla variabile x un valore, eseguendo l'istruzione di assegnazione python del tipo:

```
x=valore
```

Per eseguire una qualunque istruzione Python, la classe PythonEngine mette a disposizione il metodo Execute, con diversi overload. In questo caso, basterà creare una stringa "x=valore", dove valore sarà di volta in volta un diverso elemento del dominio, e poi dare in ingresso la stessa stringa al metodo PythonEngine.Execute.
Assegnato il valore alla variabile x si potrà procedere con il valutare la funzione, semplicemente invocando il metodo EvaluateAs, che valuta un'espressione Python restituendo un valore di un

tipo predeterminato. In definitiva tutto ciò che serve è un metodo C# come il seguente:

```
public double EvaluateFunction(string function,
                    string varName, double varValue)
{
        try
{
            engine.Execute(varName + "=" + varValue);
            return engine.EvaluateAs<double>(function);
        }
        catch
        {
            throw;
        }
}
```

Supponendo che la funzione sia rappresentata dalla stringa "x*x+1", e di volerla valutare nel punto di ascissa x=0, si invocherà il metodo EvaluateFunction così:

```
double y=EvaluateFunction("x*x+1","x",0);
```

In questo caso y assumerà il valore 1. Per calcolare il valore della funzione su tutti i punti dell'intervallo scelto, basterà un ciclo for, e per comodità si conserveranno le coppie di punti che costituiscono il grafico della funzione in un oggetto List<Point>. Se min e max rappresentano rispettivamente l'estremo inferiore e quello superiore del dominio di x, e lo step che vogliamo utilizzare è di 0.05, il ciclo sarà fatto come segue:

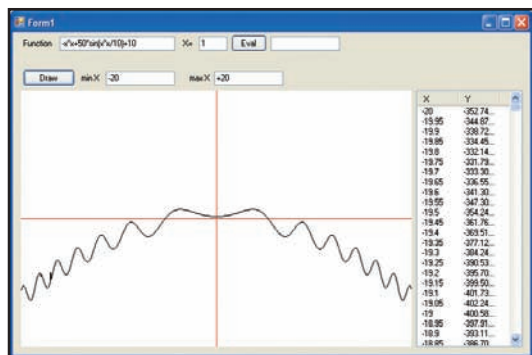



Fig. 3: **Il disegno di una funzione ad una variabile**

```
python=new IronPythonHelper();
for (double x = min; x < max; x += 0.05)
{
    y = python.EvaluateFunction
                    (txtExpression.Text, "x", x);
    points.Add(new PointF((float)x,(float)y));
}
```

A questo punto non resta che trasformare i valori reali in valori compatibili con le coordinate dello schermo, ed in particolare del Panel utilizzato e disegnare una spezzata formata dall'unione di tutti i punti ottenuti. Lasciamo perdere la spiegazione nei dettagli, che esula dallo scopo dell'articolo, mostrando solo il codice C#:

```
private void TranslatePoints()
{
        float incr = ((float) panel1.Width) /
                                        points.Count;
        pointsArray = new PointF[points.Count];
        PointF pt;
        PointF ptO=new
            PointF(panel1.Width/2,panel1.Height/2);
        for (int i = 0; i < points.Count; i++)
        {
            pt = new PointF();
            pt.X = i*incr;
          pt.Y = (ptO.Y-
              yscale*((float)points[i].Y)/panel1.Height);
          pointsArray[i]=pt;
        }
}
private void panel1_Paint(object sender,
                                    PaintEventArgs e)
{
        Graphics gfx = e.Graphics;
        gfx.DrawLine(Pens.Red,new
                        Point(panel1.Width/2,0),new
                Point(panel1.Width/2,panel1.Height));
        gfx.DrawLine(Pens.Red, new Point(0,
            panel1.Height / 2), new Point(panel1.Width,
                                panel1.Height / 2));
        if (pointsArray != null && pointsArray.Length >
                                                    0)
        {
          gfx.DrawLines(Pens.Black,pointsArray);
        }
}
```

L'applicazione completa e funzionante è mostrata in figura 3, mentre disegna una funzione abbastanza complessa.

Antonio Pelleriti

L'AUTORE

Antonio Pelleriti è ingegnere informatico e si occupa di .NET sin dalla prima versione beta. Potete rivolgere domande di chiarimenti o ulteriori richieste all'autore sul forum di IoProgrammo (forum.ioprogrammo.it) o sul sito www.dotnetarchitects.it.

INSTALLAZIONE DEI PREREQUISITI

È possibile effettuare il debug degli script IronPython da Visual Studio. Utilizzando il comando File->Open->Project/Solution … si apra il file ipy.exe. Facendo clic con il destro su ipy.exe nel Solution Explorer si selezioni Properties. Nella finestra delle proprietà bisogna riempire i due campi Command Arguments, inserendo il nome del file da debuggare e Working Directory, inserendo il path della directory in cui si trova il file. Adesso si salvi la soluzione. Per debuggare il file, basta aprirlo con il comando File->Open, impostare i breakpoint e premere F5 per far partire il debug.

SOFTWARE INTERNAZIONALE

IN UN EPOCA IN CUI LE DISTANZE SI SONO ORMAI ANNULLATE, È IMPORTANTE PRODURRE APPLICAZIONI FRUIBILI IN OGNI PARTE DEL MONDO. IL PRINCIPIO BASE È LA LOCALIZZAZIONE. VEDIAMO COME SCRIVERE CODICE MULTILINGUA

A distanza di alcuni anni dalla nascita del World Wide Web e dall'affermarsi della comunicazione globale, la sensazione ancora viva in molti è quella di "avere il mondo in casa e di poter essere in ogni parte del mondo con un semplice click". Per sfruttare al massimo questa potenzialità, chi realizza siti web o applicazioni e intende farne conoscere il contenuto in uno spazio geografico quanto più ampio possibile deve disporre di strumenti che rendano i suoi prodotti "universali", comprensibili appunto "in ogni parte del mondo". Spiegare quali siano questi strumenti è l'obiettivo del presente articolo.

GLOBALIZATION E LOCALIZATION

Globalizzazione e Localizzazione sono due facce della stessa medaglia e costituiscono la base del processo di internazionalizzazione delle applicazioni. Nello specifico per globalizzazione si intende il processo con cui si prepara l'applicazione a supportare diversi linguaggi. La localizzazione si occupa invece della traduzione dell'applicazione nella lingua desiderata.

LOCALIZZIAMO I CONTENUTI

Prima di entrare nel vivo, proviamo ad identificare alcuni dei principali elementi di un'applicazione che potrebbero necessitare di una traduzione.

- *immagini*: alcune immagini di un'applicazione potrebbero contenere del testo che dovrà essere localizzato;
- *testi statici*: anche label, descrizioni testuali statiche, tooltip devono essere tradotte nelle lingue supportate dalla nostra applicazione;
- *testi da database*: le applicazioni moderne fanno largo uso di contenuti provenienti da database, come, ad esempio, un catalogo in cui sono contenute le caratteristiche di alcuni prodotti.
- *date, numeri*: anche se queste informazioni non devono essere "tradotte", andranno, comunque, visualizzate in modo diverso in base alla lingua di destinazione. Pensiamo, al formato delle date: ad esempio il "20 ottobre 2006" sarà scritto 20/10/06 per la lingua italiana, 10/20/06 per l'inglese americano.

FACCIAMOLO CON WEBFORMS

Costruiamo una scheda che legge i dettagli di un prodotto da una tabella di database e li visualizza in una pagina web. E' un semplice esempio che ci permette comunque di analizzare tutti gli aspetti della localizzazione.
Creiamo come prima cosa il database e la tabella che conterrà le informazioni sui prodotti. Memorizziamo nome, descrizione, prezzo e data disponibilità. La chiave della tabella sarà composta dalla coppia idProdotto-lingua

```
CREATE TABLE [dbo].[Prodotti](
        [idProdotto] [int] NOT NULL,
        [lingua] [char](5) COLLATE
                Latin1_General_CI_AS NOT NULL,
        [nome] [varchar](100) COLLATE
                Latin1_General_CI_AS NOT NULL,
        [descrizione] [text] COLLATE
                Latin1_General_CI_AS NOT NULL,
        [prezzo] [money] NOT NULL,
        [disponibile_dal] [datetime] NOT NULL,
 CONSTRAINT [PK_Prodotti] PRIMARY KEY CLUSTERED
(
        [idProdotto] ASC,
        [lingua] ASC
```

```
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

Table - dbo.Prodotti | CRI\MYSQLEXP...SQLQuery1.sql | Summary

	idProdotto	lingua	nome	descrizione	prezzo	disponibile_dal
	1	en-US	Product one	Description for product one	100,0000	10/10/2006 0.00.00
	1	it-IT	Prodotto uno	Descrizione prodotto uno	100,0000	10/10/2006 0.00.00
	1	fr-FR	Produit un	Description pour le produi...	100,0000	10/10/2006 0.00.00
	1	de-DE	Produkt ein	Beschreibung für Produkt...	100,0000	10/10/2006 0.00.00
▶*	NULL	NULL	NULL	NULL	NULL	NULL

Fig. 1: **La tabella prodotti**

Inseriamo quindi un prodotto di esempio
Iniziamo il processo di globalizzazione preparando la nostra applicazione affinché possa gestire le lingue.
Creiamo innanzitutto uno "User Control" in cui

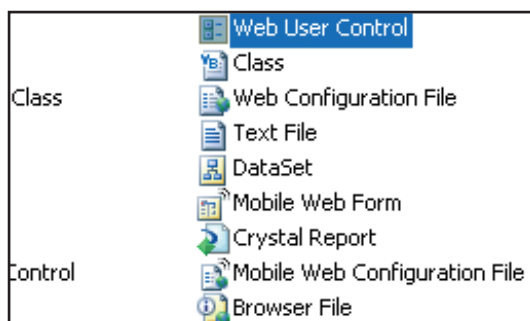



Fig. 2: **Il controller per la selezione della lingua**

inseriremo i pulsanti per cambiare le lingue. Clicchiamo con il tasto destro del mouse sul progetto e selezioniamo "Add New Item..." quindi "Web User Control", chiamiamo il file che stiamo creando "LanguageController.ascx".
All'interno, inseriamo i pulsanti per cambiare la lingua attiva ed il codice necessario per notificarne il cambiamento all'applicazione.

Il file *LanguageController.ascx*

```
<%@ Control Language="VB" AutoEventWireup="false"
        CodeFile="LanguageController.ascx.vb"
        Inherits="LanguageController" %>
<div style="white-space:nowrap;">
<asp:ImageButton ToolTip="Italiano" ID="ibn_ita"
        runat="server" ImageUrl="~/images/flag_ita.jpg"
        /> 
<asp:ImageButton ToolTip="English" ID="ibn_eng"
        runat="server" ImageUrl="~/images/flag_eng.jpg"
        /> 
<asp:ImageButton ToolTip="Français" ID="ibn_fra"
        runat="server" ImageUrl="~/images/flag_fra.jpg"
        /> 
<asp:ImageButton ToolTip="Deutsch" ID="ibn_ger"
        runat="server" ImageUrl="~/images/flag_ger.jpg"
        />  
</div>
```

Il file *LanguageController.ascx.cs*

```
Partial Class LanguageController
    Inherits System.Web.UI.UserControl
    Protected Sub ibn_ita_Click(ByVal sender As
Object, ByVal e As System.Web.UI.ImageClickEventArgs)
        Handles ibn_ita.Click
        LanguageChange("it-IT")
    End Sub
    Protected Sub ibn_eng_Click(ByVal sender As
Object, ByVal e As System.Web.UI.ImageClickEventArgs)
        Handles ibn_eng.Click
        LanguageChange("en-US")
    End Sub
    Protected Sub ibn_fra_Click(ByVal sender As
Object, ByVal e As System.Web.UI.ImageClickEventArgs)
        Handles ibn_fra.Click
        LanguageChange("fr-FR")
    End Sub
    Protected Sub ibn_ger_Click(ByVal sender As
Object, ByVal e As System.Web.UI.ImageClickEventArgs)
        Handles ibn_ger.Click
        LanguageChange("de-DE")
    End Sub
    Private Sub LanguageChange(ByVal culture As
        String)
        Response.Cookies("culture").Value =
            culture
        Response.Cookies("culture").Expires
            = DateTime.Now.AddMonths(120)
Response.Redirect(Request.UrlReferrer.ToString())
    End Sub
End Class
```

Il controller non fa altro che scrivere un cookie con il nome della lingua selezionata e ricaricare la pagina.
Utilizziamo quindi il *Global.asax* per definire la lingua che stiamo utilizzando. Occorre importare due namespace:

```
<%@ Import Namespace="System.Globalization" %>
<%@ Import Namespace="System.Threading" %>
```

Nel metodo *Application_BeginRequest*, leggiamo il cookie che è stato scritto; creiamo un oggetto **CultureInfo** in base alla lingua selezionata e di conseguenza impostiamo la lingua del Thread corrente tramite ***CurrentClture*** e ***Current UICulture***.
Utilizziamo il metodo **Application_Begin Request** perché questo viene eseguito ad ogni richiesta di pagina. Questo ci consente di risparmiare codice in un sito che ha più pagine in quanto non dobbiamo inserire il codice per l'impostazione della lingua in tutte le pagine.

```
Sub Application_BeginRequest(ByVal sender As Object,
        ByVal e As EventArgs)
    Dim lang As CultureInfo
```

NOTA

I FILE RESOURCES

I file di risorsa (.resx) sono dei file che utilizzano XML per l'accesso ai dati e possono contenere testi, immagini, icone ed altri oggetti. Possono essere creati con Visual Studio oppure tramite un'utilità del framework chiamata resgen.exe o infine in modo programmatico utilizzando la classe ResourceWriter del namespace System.Resources. Sono molto utili per localizzare testi statici o immagini in base alla lingua selezionata.

```
        If Request.Cookies("culture") Is Nothing Then
            Response.Cookies("culture").Value =
                "it-IT"
        End If
        Try
            lang = New
              CultureInfo(Request.Cookies("culture").Value)
        Catch
            lang = New CultureInfo("it-IT")
        End Try
        Thread.CurrentThread.CurrentCulture = lang
        Thread.CurrentThread.CurrentUICulture = lang
End Sub
```

In particolare la riga

```
lang = New
       CultureInfo(Request.Cookies("culture").Value)
```

crea l'oggetto **CoultureInfo** in base alla lingua selezionata dall'utente.
Mentre le seguenti righe:

```
Thread.CurrentThread.CurrentCulture = lang
Thread.CurrentThread.CurrentUICulture = lang
```

assegnano l'oggetto CultureInfo al Thread corrente.
CurrentCulture specifica la cultura utilizzata per visualizzare informazioni e dati , viene usata per l'ordinamento delle stringhe, la visualizzazione di date, numeri e valute.
CurrentUICulture permette di specificare la cultura per l'interfaccia utente (User Interface) e viene utilizzata quando si ricercano le informazioni nei file di risorsa.
A questo punto, siamo pronti per creare la pagina che visualizza i prodotti.

```
<%@ Page Language="VB" AutoEventWireup="false"
  CodeFile="Default.aspx.vb" Inherits="_Default" %>
<%@ Register Src="LanguageController.ascx"
  TagName="LanguageController" TagPrefix="uc1" %>
[omissis...]
<table width="100%">
<tr>
        <td colspan="2"><asp:Image
           ID="img_titolo" runat="server" /></td>
        <td align="right" style="width: 184px"
                                   valign="top">
            <uc1:LanguageController
        ID="LanguageController1" runat="server" />
        </td>
</tr>
</table>
```

Omettiamo parte del codice che potrete comunque trovare nel CD allegato; il file in questione è "Default.aspx" nella cartella "LocWeb".
Nella pagina è già presente il controllo che permette di cambiare la lingua corrente.
Gli altri componenti sono: un'immagine "img_titolo" che conterrà il titolo della pagina, le Label con le descrizioni e alcuni TextBox che conterranno i dati letti dal database.
Occupiamoci come prima cosa di localizzare le immagini.
Creiamo una cartella chiamata "images" e, al suo interno, tante cartelle quante sono le lingue che intendiamo supportare, chiamandole con la sigla della lingua in questione. Nel nostro esempio ne creiamo quattro, nello specifico *"it-IT", "en-US", "fr-FR" e "de-DE"*.
A questo punto, creiamo le quattro immagini con il titolo nelle diverse lingue e salviamole nelle cartelle della lingua corrispondente con nome "header.gif".
Fatto questo, associare le immagini corrette alla lingua è abbastanza semplice.
Nel file *Default.aspx.cs* dichiariamo una variabile globale

```
Private lang As CultureInfo
```

che rappresenta la lingua selezionata e creiamo la procedura **Page_PreInit**

```
Protected Sub Page_PreInit(ByVal sender As Object,
    ByVal e As System.EventArgs) Handles Me.PreInit
        lang = Thread.CurrentThread.CurrentCulture
        'Immagine
        img_titolo.ImageUrl = "images/" &
                        lang.Name & "/header.gif"
End Sub
```

La procedura non fa altro che leggere il valore di **CurrentCulture** e assegnare l'immagine corretta utilizzando la proprietà **Name** di lang.
Per localizzare le etichette con le descrizioni ed altri elementi della pagina utilizziamo uno strumento molto comodo messo a disposizione da .Net: i file di risorsa.
Questi file conterranno le informazioni sugli elementi della pagina nelle varie lingue.
Clicchiamo con il tasto destro del mouse sul progetto e selezioniamo "**Add New Item...**", quindi "**ResourceFile**". Chiamiamo il file "Resource.resx"; sarà il file per la lingua di default.
Se apriamo Resource.resx in Visual Studio, questo si presenterà come una tabella. Notiamo due colonne: **Name** e **Value**. Nella prima metteremo i nomi delle risorse (a cui accederemo poi da codice per localizzare i contenuti), nella seconda inseriremo il valore della risorsa.

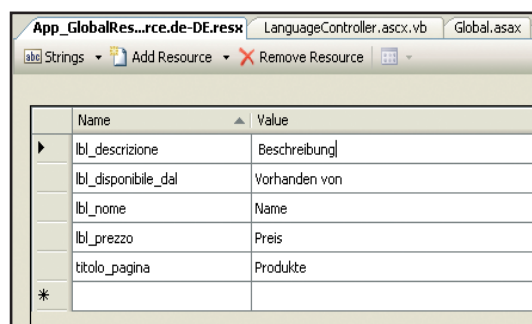



Fig. 2: ***Il file di risorse per la lingua tedesca***

Ripetiamo l'operazione per le altre quattro lingue avendo l'accortezza di nominare i file con "Resource.LINGUA.resx". Per il francese, il nome sarà, ad esempio, "Resource.fr-FR.resx".
Trovate i file di risorsa utilizzati nel CD allegato.
A questo punto, siamo in grado di localizzare i testi delle etichette. Per farlo, utilizziamo ancora una volta la procedura Page_PreInit vista prima. Inseriamo le seguenti righe in coda al codice già scritto.

```
'Etichette
lbl_nome.Text = Resource.lbl_nome
lbl_descrizione.Text = Resource.lbl_descrizione
lbl_prezzo.Text = Resource.lbl_prezzo
lbl_disponibile_dal.Text =
                         Resource.lbl_disponibile_dal
'Titolo Pagina
Page.Title = Resource.titolo_pagina
```

Abbiamo utilizzato la sintassi **nome_label.Text = Resource.nome_risorsa**
Così facendo, i testi delle *Label* ed il *Title* della pagina verranno letti dinamicamente dai file di risorsa in base alla lingua selezionata.
Possiamo adesso occuparci di leggere i contenuti da database.
Creiamo una classe "**Prodotto**" per *mappare* i campi della tabella prodotti. Importiamo due namespace: System.Data:SqlClient per l'accesso a SQL Server e System.Globalization per la localizzazione.

```
Imports Microsoft.VisualBasic
Imports System.Data.SqlClient
Imports System.Globalization

Public Class Prodotto
        Private _nome As String
        Private _descrizione As String
        Private _prezzo As Double
        Private _disponibile_dal As DateTime
        Public Sub New(ByVal idProdotto As Integer,
           ByVal lang As CultureInfo, ByVal Conn As
                                     SqlConnection)
                Try
                        Conn.Open()
                        Dim Sql As String = ""
                                                 &
                        "SELECT " & _
                        "  nome, descrizione,
                     prezzo, disponibile_dal" & _
                        " FROM Prodotti " & _
                        " WHERE lingua =
                                      @lingua" & _
                        " AND idProdotto =
                                     @idProdotto"
                        Dim Comm As
             SqlCommand = New SqlCommand(Sql, Conn)
Comm.Parameters.Add(New SqlParameter("@lingua",
                                       lang.Name))
Comm.Parameters.Add(New
           SqlParameter("@idProdotto", idProdotto))
                                   Dim Areader As
           SqlDataReader = Comm.ExecuteReader()
                                              Try
                                   If
                                 Areader.Read Then
Me._nome = Areader("nome")
Me._descrizione = Areader("descrizione")
Me._prezzo = Areader("prezzo")
Me._disponibile_dal = Areader("disponibile_dal")
                                   End If
                        Finally
Areader.Close()
                        End Try
                Finally
                        Conn.Close()
                End Try
        End Sub
        Public ReadOnly Property nome() As String
                Get
                        Return _nome
                End Get
        End Property
        Public ReadOnly Property descrizione() As
                                           String
                Get
                        Return _descrizione
                End Get
        End Property
        Public ReadOnly Property prezzo() As
                                           Double
                Get
                        Return _prezzo
                End Get
        End Property
        Public ReadOnly Property disponibile_dal() As
                                         DateTime
                Get
                        Return _disponibile_dal
                End Get
        End Property
End Class
```

La classe ha un costruttore *overloaded* che legge dal database i dettagli del prodotto in base alla lingua e li salva in dei campi privati accessibili tramite delle proprietà di sola lettura.
A questo punto, non ci resta che istanziare la classe nel *Page_Load* ed il gioco è fatto...

```
Protected Sub Page_Load(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Me.Load
            If Not IsPostBack Then
                Dim idProdotto =
                    Request.QueryString("idProdotto")
                If idProdotto = "" Then
                    idProdotto = 1
                End If
                Try
                    Dim Conn As
                        SqlConnection = New
SqlConnection(ConfigurationManager.ConnectionStrin
            gs("ConnectionString").ConnectionString)
                    Dim P As Prodotto =
                        New Prodotto(idProdotto, lang, Conn)
                txt_nome.Text = P.nome
                    txt_descrizione.Text =
                        P.descrizione
                    txt_prezzo.Text =
                        P.prezzo.ToString()
                    txt_disponibile_dal.Text
                        = P.disponibile_dal.ToString()
                Catch ex As Exception
Response.Redirect("ErrorPage.aspx")
                End Try
            End If
End Sub
```

Resta un'ultima cosa da fare. Date e numeri hanno modalità di visualizzazione diversa in base alla lingua, basti pensare al separatore dei decimali che può essere il punto oppure la virgola. Per visualizzare correttamente questi ultimi, creiamo una classe Globalizer che contiene due metodi FormatData e FormatNumero.

```
Imports Microsoft.VisualBasic
```

LE "CULTURE"

Per poter specificare in modo univoco una cultura, è stato introdotto uno standard chiamato RFC 1766 che definisce codici univoci per ciascuna di esse. Questi sono formati da due parti separate da un trattino: la prima (in minuscolo) rappresenta la lingua, la seconda (in maiuscolo) specifica il paese. Ecco alcuni esempi
- it-IT: italiano – Italia
- en-GB: inglese – Regno unito
- en-US: inglese – Stati Uniti
Per una lista completa si può consultare l'MSDN
http://msdn2.microsoft.com/en-us/library/system.globalization.cultureinfo.aspx

```
Imports System.Globalization
Public Class Globalizer
        Public Shared Function FormatData(ByVal input
        As Object, ByVal lang As CultureInfo) As String
            Try
                Dim tmp As DateTime =
                    Convert.ToDateTime(input)
                Dim dtfi As
                    DateTimeFormatInfo = lang.DateTimeFormat
                Return
                    tmp.ToString(dtfi)
            Catch ex As Exception
                Return ""
            End Try
        End Function
        Public Shared Function FormatNumero(ByVal
    input As Object, ByVal lang As CultureInfo , ByVal
                decimali As Integer) As String
            Try
                Dim tmp As Double =
                    Convert.ToDouble(input)
                Dim nfi As
                    NumberFormatInfo = lang.NumberFormat
nfi.NumberDecimalDigits
                        = decimali
                Return tmp.ToString
                    ("N", nfi)
            Catch ex As Exception
                Return ""
            End Try
        End Function
End Class
```

A questo punto, ci basta utilizzare le due funzioni.

Fig. 4: *La scheda prodotto*

```
txt_prezzo.Text = Globalizer.FormatNumero(P.prezzo,
                                            lang, 2)
txt_disponibile_dal.Text =
```

```
Globalizer.FormatData(P.disponibile_dal, lang)
```

E lanciare l'applicazione!

FACCIAMOLO CON WINFORMS

Il processo di localizzazione è leggermente diverso per quanto riguarda WinForms.
Creiamo un nuovo progetto Windows ed impostiamo il layout della form aggiungendo infine i controlli per la visualizzazione dei testi e della grafica: una **PictureBox** per il logo, le **Label** per le descrizione e le **TextBox** per i valori recuperati dal database.

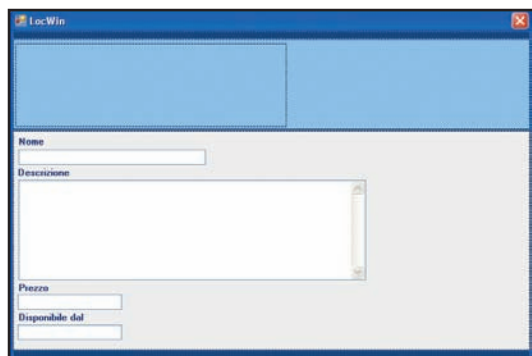



Fig. 5: Impostiamo il layout

Impostiamo quindi un attributo della form "**Localizable = true**".
Questo attributo permette a Visual Studio di generare i file di risorse in modo automatico. Per farlo, è sufficiente cambiare l'attributo "**Language**" nella lingua che vogliamo gestire e

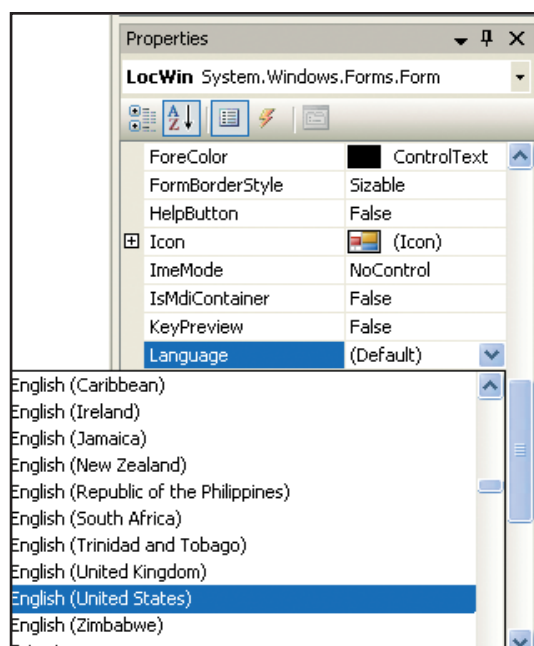



Fig. 6: Selezioniamo la lingua da creare

modificare i valori da localizzare.
Selezioniamo come lingua "**English** (United States)" e cambiamo l'immagine del logo ed i testi della Label. Vedremo che viene creato in automatico il file "**LocWin.en-US.resx**".
Ripetiamo l'operazione per le altre lingue che vogliamo impostare.
A questo punto, abbiamo gestito le risorse statiche; per quelle dinamiche possiamo utilizzare i file "Globalizer.vb" e "Prodotto.vb" creati per l'applicazione web.
Importiamoli nel progetto cliccando sul nome dello stesso con il tasto destro del mouse e selezionando "Add Existing Item".
Nella form effettuiamo l'overload del costruttore di default per consentire alla form di essere caricata in base ad una specifica lingua.
Il codice del file LocWin.bv è il seguente

Fig. 7: Il nostro "launcher"

```
Imports System.Globalization
Imports System.Threading
Imports System.Data.SqlClient
Public Class LocWin
    Dim lang As CultureInfo
    Public Sub New(ByVal culture As String)
        ImpostaLingua(culture)
        InitializeComponent()
        Dim idProdotto As Integer = "1"
        Try
            Dim Conn As
SqlConnection = New SqlConnection("Data
Source=CRI\MYSQLEXPRESS;Initial
Catalog=Internationalization;Integrated
Security=True")Dim P As Prodotto = New
Prodotto(idProdotto, lang, Conn)
            txt_nome.Text =P.nome
            txt_descrizione.Text =
P.descrizione
            txt_prezzo.Text =
Globalizer.FormatNumero(P.prezzo, 2, lang)
        txt_disponibile_dal.Text=
Globalizer.FormatData(P.disponibile_dal, lang)
        Catch ex As Exception
MessageBox.Show("Error")
        End Try
    End Sub
    Private Sub ImpostaLingua(ByVal c As
```

```
                                   String)
              lang = New CultureInfo(c)
Thread.CurrentThread.CurrentCulture = lang
Thread.CurrentThread.CurrentUICulture = lang
        End Sub
End Class
```

La logica è la stessa che abbiamo utilizzato nella costruzione della pagina web.
Manca un ultimo passo prima di completare l'applicazione. Creiamo una nuova WinForm "Launcher" che conterrà le bandierine rappresentanti le lingue a cui e sarà associato il codice per istanziare la form in una determinata lingua.
Ecco il codice relativo al launcher

```
Imports System.Globalization
Imports System.Threading
Public Class Launcher
        Private Sub btn_ita_Click(ByVal sender As
          System.Object, ByVal e As System.EventArgs)
          Handles btn_ita.ClickDim lw As LocWin = New
                                      LocWin("it-IT")
              lw.ShowDialog()
        End Sub
        Private Sub btn_eng_Click(ByVal sender As
          System.Object, ByVal e As System.EventArgs)
        Handles btn_eng.Click Dim lw As LocWin = New
                                      LocWin("en-US")
              lw.ShowDialog()
        End Sub
        Private Sub btn_fra_Click(ByVal sender As
          System.Object, ByVal e As System.EventArgs)
         Handles btn_fra.Click Dim lw As LocWin = New
                                      LocWin("fr-FR")
              lw.ShowDialog()
        End Sub
        Private Sub btn_ted_Click(ByVal sender As
          System.Object, ByVal e As System.EventArgs)
         Handles btn_ted.Click Dim lw As LocWin = New
                                      LocWin("de-DE")
              lw.ShowDialog()
        End Sub
End Class
```

L'AUTORE

Carmelo Scuderi è ingegnere informatico. Si occupa di sviluppo web-based in ambiente .Net per una software house di Milano. Gestisce un sito ricco di script e manuali per chi si affaccia al mondo della programmazione web (www.morpheusweb.it).

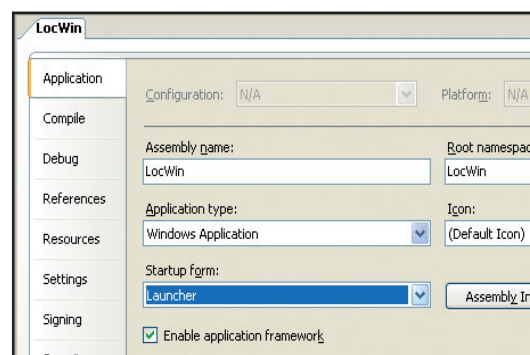



Fig. 8: ***Cambiamo la form di avvio***

Al click su una bandiera corrisponde la creazione di una nuova istanza di LocWin a cui viene passata l'informazione sulla lingua da utilizzare.
A questo punto, non ci resta che cambiare la form di partenza dell'applicazione. Per farlo, clicchiamo con il tasto destro del mouse sul progetto e selezioniamo "Properties". Dalla sezione "Application" impostiamo "Launcher" come valore per l'opzione "Startup form".
Abbiamo finito! Possiamo eseguire l'applicazione.

Fig. 9: ***Il risultato finale***

CONCLUSIONI

Con i nostri esempi abbiamo dimostrato come sia relativamente semplice realizzare delle applicazioni multilingua e abbiamo voluto sottolineare come il buon esito dell'operazione sia strettamente vincolato all'individuazione corretta, a priori, degli elementi che dobbiamo tradurre. Una volta strutturata l'applicazione in modo che possa essere multilingua, ossia una volta globalizzata, localizzarla in 3 lingue piuttosto che 4 o 5 non comporta un ulteriore carico di lavoro. Infatti, non sarà necessario scrivere nuovo codice; l'unica cosa da fare sarà tradurre i contenuti e inserirli nel database e nei file di risorse.

Carmelo Scuderi

GLI STRUMENTI MESSI A DISPOSIZIONE DA .NET

**Nel framework .Net è presente il namespace System.Globalization destinato alla localizzazione delle informazioni.
La classe principale è CultureInfo che rappresenta una cultura specifica e ne definisce la visualizzazione di numeri, valute, date ed altro.
La classe RegionInfo fornisce, invece, informazioni sul paese.
Altre classi utili sono NumberFormatInfo per la formattazione dei numeri e DatetimeFormatInfo per la formattazione delle date.
Informazioni sul Namespace System.Globalization su:
http://msdn2.microsoft.com/it-it/library/system.globalization.aspx**

CREARE REPORT RTF CON ASP.NET 2.0

LA CREAZIONE DI REPORTISTICA È SEMPRE STATA DEMANDATA A PACCHETTI ESTERNI. MA CON RTF SI PUÒ IMPLEMENTARE IN POCHI E SEMPLICI PASSI. SOPRATTUTTO SI PUÒ UTILIZZARE UN FORMATO UNIVERSALE ALMENO QUANTO PDF

La reportistica è alla base di tutte le applicazioni che manipolano dati, poiché consente di visualizzarli in maniera più semplice. Le tecniche in questo ambito sono differenti e prevedono l'utilizzo di strumenti che automatizzano gran parte del lavoro, come **Crystal Report**, piuttosto che soluzioni fatte in casa. Queste ultime hanno il vantaggio di essere più flessibili, garantire un'adattabilità più semplice ed un utilizzo dei formati che si preferisce. Lo scopo di questo articolo è quello di analizzare a fondo le caratteristiche di un sistema di reportistica, costruito da zero, che abbia come formato utilizzato RTF.

Fig. 1: Pagina di autenticazione

PERCHÉ LA SCELTA DI RTF

RTF (**Rich Text Format**) è un formato aperto, disponibile su larga scala, le cui specifiche sono note, aperte e facilmente implementabili, poiché è interamente basato su testo in formato ASCII circondato da tag.

Il vantaggio principale di RTF sta nell'immediatezza con il quale è possibile generare documenti dalla struttura anche molto complessa, poiché consiste di poche regole, che possono essere facilmente implementate. Al pari di **PDF**, poi, RTF è un formato largamente diffuso, praticamente su tutte le piattaforme. L'idea alla base di questo articolo è dunque quella di creare un sistema in grado di generare i giusti tag RTF, così da avere alla fine un file che contenga esattamente l'output desiderato, contornato da una serie di informazioni di tipo stilistico, come la dimensione del font, il grassetto o l'italico, aggiunte in maniera del tutto naturale.

Alla fine di questo articolo, dopo aver analizzato alcune tra le possibili soluzioni, si arriverà a quella più complessa, che prevede la creazione di una class library ad hoc per gestire la creazione di documenti in questo formato.

LA VIA SEMPLICE: CON UN TEMPLATE

La prima tipologia di soluzione, si basa per intero sulla creazione di un template, all'interno del quale inserire dei segnaposto, con la successiva sostituzione di questi ultimi con i dati effettivamente recuperati dalla nostra fonte.

Un approccio del genere risulta vincente, oltre che comodo, nel tipico sistema di fatturazione (o gestionale), dove la fattura è appunto contraddistinta dall'avere sempre lo stesso identico formato.

Questa tecnica, poi, ha il vantaggio di rendere la creazione del template estremamente semplice, perché è sufficiente aprire Word, o un editor di RTF simile, per creare interfacce anche complesse, dal punto di vista grafico, che includano loghi o formattazioni particolari.

L'uso della classi della BCL del .NET Framework farà poi il resto, grazie alla possibilità di aprire il template, provvedere alla sostituzione del testo e salvare il risultato finale, piuttosto che mostrarlo direttamente a video, consentendo all'utente anche di stamparlo.

In questo caso l'implementazione è abbastanza semplice e consiste nell'utilizzo della classe statica **File**, contenuta nel namespace **System.IO**. Questa classe ha un metodo, sempre statico, di nome **ReadAllText**, che è stato introdotto a partire dalla versione 2.0 e condensa al proprio interno l'utilizzo di uno **StreamReader**, l'oggetto utilizzato a partire dalla versione 1.0 per leggere informazioni da uno stream creato a partire da un file.

Una volta caricato in memoria, si può procedere alla

sostituzione attraverso il metodo **Replace** di String, piuttosto che utilizzando una **Regular Expression**, se le tipologie di sostituzioni sono più complesse.
Nel caso preso in esame si è optato per l'utilizzo di un formato per i segnaposti molto immediato, nella forma *#segnaposto#*, così da facilitare il compito sia in fase di creazione del template, che per quanto riguarda il codice, che diventa davvero semplice, come si può notare da quanto riportato in basso:

```
// carico il template
string template =
            File.ReadAllText(Server.MapPath("template.rtf"),
      System.Text.Encoding.GetEncoding("iso-8859-15"));
// sostituzione del testo
template = template.Replace("#firstname#",
                                        FirstNameBox.Text);
template = template.Replace("#lastname#",
                                        LastNameBox.Text);
// output a video
Response.Clear();
Response.ContentType="text/rft";
Response.AppendHeader("Content-Disposition",
                          "attachment; filename=report.rtf");
Response.Write(template);
Response.End();
```

L'esecuzione di una pagina con questo codice si traduce nell'avere un documento RTF, che sarà possibile aprire, ad esempio, direttamente con Word.
Questo approccio è molto potente, anche per via della facilità con cui è possibile personalizzare l'output, ma mostra tutti i suoi limiti quando è necessario estrarre quantità di informazioni maggiori, ad esempio prendendo come fonte dati una query effettuata su un database. È in questi casi, invece, che una soluzione più complessa mostra tutti i suoi vantaggi.

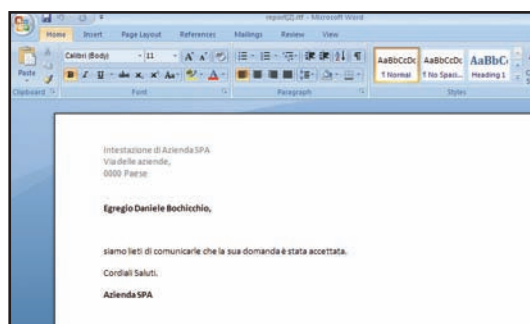



Fig. 2: ***Il risultato***

IL FORMATO RTF

Le specifiche RTF sono disponibili all'indirizzo http://support.microsoft.com/kb/q86999/.
Come già detto, è possibile implementare proprie soluzioni che utilizzino questo formato senza essere vincolati a royalty o limitazioni.
Nel caso di una tabella, in realtà, il formato di RTF è tutt'altro che simile ad HTML, dove si ragiona con righe e colonne.
Basta creare un semplice file con Word, come esempio, ed aprirlo con notepad per rendersene conto. Ripulito da tutto il codice che RTF aggiunge per la formattazione (in realtà molto verboso), il codice necessario a creare una tabella di due righe si riduce a questo:

```
\trowd\trautofit1
\intbl
\cellx1
\cellx2
\cellx3
{\rtlch\fcs1 \af31507 \ltrch\fcs0
A1\cell A2\cell A3\cell
\row}
{\trowd\trautofit1
\intbl
\cellx1
\cellx2
\cellx3}

\trowd\trautofit1
\intbl
\cellx1
\cellx2
\cellx3
{\rtlch\fcs1 \af31507 \ltrch\fcs0
B1\cell B2\cell B3\cell
\row}
{\trowd\trautofit1
\intbl
\cellx1
\cellx2
\cellx3}
```

La tabella è iniziata dal tag **\trowd**, che insieme a **\trautofit1** ha l'effetto di non richiedere la dichiarazione delle dimensioni delle singole colonne, cosa che facilita di molto la creazione di una funzione generica in grado di generare una tabella qualsiasi a fronte del risultato di una query, contenuta all'interno di un oggetto di tipo DataTable.
Il contenuto vero e proprio della riga della tabella è racchiuso all'interno delle parantesi {} che contengono i tag **\cell**, che rappresentano la fine del contenuto della cella stessa.
Il tag **\intbl** indica invece che il contenuto che segue è un paragrafo di tipo tabellare, mentre le varie **\cellxn** servono ad indicare le colonne da aggiungere alla tabella.
Il tag **\row** è praticamente il tag di chiusura di \trowd, con l'effetto di chiudere dunque la riga.
Per ogni riga va ripetuto questo codice, così che in pratica la dimensione della tabella è indicata dalla

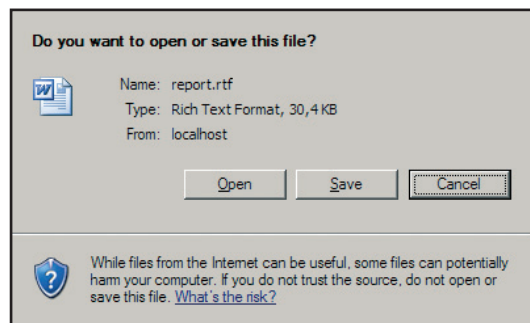



Fig. 3: **I controlli di sistema**

tipologia di informazioni contenute all'interno dei tag generati. La creazione dunque di una classe in grado di generare output direttamente in formato RTF può partire isolando le casistiche più importanti. Ad esempio, un testo in **grassetto** è incluso all'interno di una sequenza {\b testo }, mentre il corsivo è semplicemente {\i testo }. Basta cominciare ad implementare un po' di metodi che scrivano all'interno di uno **StringBuilder**, se invocati, il contenuto passato come argomento. E tenere presente, allo stesso tempo, che il documento deve avere le informazioni sul formato, subito dopo il tag \rtf1, che serve per specificare al programma che andrà ad aprire il documento generato che si tratta effettivamente di un documento RTF. Nel caso specifico, la struttura base del documento è fatta così:

```
{\rtf1\ansi
\margl1134\margr1134\margt1418\margb1134
Contento del file
}
```

La seconda riga imposta i margini del documento, facendo in modo che nel formato A4 vengano rispettati correttamente.
Partendo da queste funzionalità di base, non ci resta che implementare il codice legato alla creazione della tabella, che deve consistere di un po' di cicli, come in questo esempio:

```
// ciclo su righe per estrarre il contenuto
for (int k = 0; k < dt.Rows.Count; k++)
{
        // stili
        text.Append("\\trowd\\trautofit1\\intbl");
        for (int j = 0; j < dt.Columns.Count; j++)
                text.Append("\\cellx" + (j+1));
        // contenuto
        text.Append("{");
        for (int j = 0; j < dt.Columns.Count; j++)
        {
text.Append(dt.Rows[k][j].ToString());
                text.Append("\\cell ");
        }
        text.Append("}");
        // stili 2
        text.Append("{");
        for (int j = 0; j < dt.Columns.Count; j++)
                text.Append("\\cellx" + (j+1));
        text.Append("\\row }");
}
```

Si è scelto di utilizzare una **DataTable** perché garantisce che i dati contenuti all'interno possano essere prelevati da qualsiasi fonte, da un database o anche da un oggetto custom, garantendo una certa omogeneità di funzionamento, grazie al fatto che è un oggetto in grado di contenere qualsiasi tipologia di informazione al proprio interno. Il risultato è quello di avere una tabella generata in maniera molto semplice, a partire da qualsiasi query.

GENERARE L'INTESTAZIONE DELLA TABELLA

Per l'intestazione si è scelto di utilizzare sempre una cella normale, con l'aggiunta del tag necessario a generare il grassetto. Il ciclo in questo caso viene effettuato sulle colonne, che attraverso la proprietà Columns la DataTable espone, con tanto di nome delle stesse. Questa caratteristica rende possibile l'aggiunta di un'intestazione generica rispetto al contenuto vero e proprio della DataTable, così da rendere praticamente universale il codice prodotto.
La parte prima e dopo il codice che segue è in realtà praticamente identica a quanto specificato per una normale cella, quello che davvero cambia è l'estrazione dei nomi delle colonne:

```
// nome delle colonne
text.Append("{");
for (int j = 0; j < dt.Columns.Count; j++)
{
        this.WriteBold(dt.Columns[j].ColumnName);
text.Append("\\cell ");
}
text.Append("}");
```

Il risultato è quello di avere ora i nomi delle colonne, a prescindere dalla fonte dati, riportate in cima alla tabella, come riferimento per i dati contenuti.

UN VIA ALTERNATIVA PER CHI USA VISUAL WEB DEVELOPER 2005 EXPRESS

L'utilizzo di VWD 2005 Express vincola al non poter utilizzare un progetto e quindi nemmeno una class library.
Dopo aver scaricato il codice associato a questo articolo è sufficiente prelevare il file RtfWriter.cs e copiarlo nella directory /App_Code/, sotto la root del sito. In questo modo la classe sarà compilata in automatico, insieme al sito web, e potrà essere utilizzata nella pagina senza alcun limite.

L'AUTORE

Daniele Bochicchio è il content manager di ASPItalia.com, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. È Microsoft ASP.NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo http://blogs.aspitalia.com/daniele/

Report degli autori

Generato mercoledì 4 ottobre 2006 alle ore 10.31.56

Last Name	First Name	Address	City	ZIP	Phone
White	Ciro	10932 Bigge Rd.	Menlo Park	94025	408 496-7223
Green	Ambrogio	309 63rd St. #411	Oakland	94618	415 986-7020
Carson	Cheryl	589 Darwin Ln.	Berkeley	94705	415 548-7723
O'Leary	Michael	22 Cleveland Av. #14	San Jose	95128	408 286-2428
Straight	Dean	5420 College Av.	Oakland	94609	415 834-2919
Smith	Meander	10 Mississippi Dr.	Lawrence	66044	913 843-0462
Bennet	Abraham	6223 Bateman St.	Berkeley	94705	415 658-9932

Fig. 4: *Una tabella RTF*

AGGIUNGERE I BORDI ALLE TABELLE

Perché il nostro sistema di reportistica possa dirsi quanto meno in grado di reggere altri sistemi analoghi, è necessario aggiungere un bordo alle celle.
Poiché in RTF in realtà una tabella è essenzialmente un paragrafo speciale, i tag da aggiungere per avere come output una tabella sono in realtà molto più complessi, dal punto di vista semantico, di quanto ad esempio preveda l'HTML, che consente di specificare la proprietà di questo tipo direttamente sulla tabella. In RTF, invece, il bordo va impostato su ogni singola cella, con l'effetto che praticamente è necessario farlo all'interno del ciclo, così da avere effetto su tutte le celle della tabella.
I bordi poi possono essere impostati in maniera indipendente l'uno dagli altri, così che il codice necessario per ogni singola cella diventa il seguente:

```
\clbrdrt\brdrs\brdrw10
\clbrdrl\brdrs\brdrw10
\clbrdrb\brdrs\brdrw10
\clbrdrr\brdrs\brdrw10
\cellx1
```

Come si può notare è necessario anteporre la definizione dello stile prima della dichiarazione stessa della cella, così che poi quando viene effettivamente riempita la cella, possa essere applicato.
È interessante notare, poi, come nel blocco di tag appena postato, il primo definisca il bordo superiore, il secondo quello di sinistra, poi quello inferiore ed infine quello di destra. Il tag \brdrw10 indica la dimensione, che è fissata in 1 twpis.
Il **twips** è un'unita di misura particolare, che vale 0,05 pollici, cioè 1,125 millimetri, ed è utilizzata in tutti i casi in cui sia necessario specificare una dimensione.
È anche possibile utilizzare misure decimali, ad esempio 05 avrebbe avuto l'effetto di aggiungere un bordo di 0,5 twips di spessore.
L'operazione va ripetuta, all'interno del ciclo, su ogni singola cella, per ogni riga.
È utile sottolineare che il bordo della tabella è regolato da altri tag, che vanno specificati prima delle definizione delle stessa:

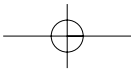
```
\trbrdrt\brdrs\brdrw10
\trbrdrl\brdrs\brdrw10
\trbrdrb\brdrs\brdrw10
\trbrdrr\brdrs\brdrw10
\trbrdrh\brdrs\brdrw10
\trbrdrv\brdrs\brdrw10
```

Gli ultimo due indicano rispettivamente il bordo orizzontale e verticale della tabella.
Messo tutto insieme, isolando il codice che genera le definizioni in un metodo a parte, così da evitare di ripetere inutilmente codice superfluo, il ciclo diventa:

```
for (int k = 0; k < dt.Rows.Count; k++)
{
        // stili
        text.Append("\\trowd\\trautofit1\\intbl");
        GenerateCellAttributes(dt);
        // contenuto
        text.Append("{");
        for (int j = 0; j < dt.Columns.Count; j++)
        {
text.Append(dt.Rows[k][j].ToString());
                text.Append("\\cell ");
}
        text.Append("}");
        // stili 2
        text.Append("{");
        text.Append("\\trowd\\trautofit1\\intbl");
        GenerateCellAttributes(dt);
        text.Append("\\row }");
}
```

Dove il metodo GenerateCellAttributes, invece, contiene le definizioni degli stili, praticamente solo relativi ai bordi, in questo modo:

```
private void GenerateCellAttributes(DataTable dt)
{
        for (int j = 0; j < dt.Columns.Count; j++)
        {
text.Append("\\clbrdrt\\brdrs\\brdrw10");
text.Append("\\clbrdrl\\brdrs\\brdrw10");
text.Append("\\clbrdrb\\brdrs\\brdrw10");
text.Append("\\clbrdrr\\brdrs\\brdrw10");
                text.Append("\\cellx" + (j + 1));
        }
}
```

A questo punto il generatore di report è pronto, non ci resta che creare una pagina in grado di renderne possibile l'interrogazione.

METTERE TUTTO INSIEME

Ed ora la parte più semplice, poiché mettere insieme il tutto è davvero il compito meno impegnativo. Una volta generato l'assembly, a partire dalla compilazione delle class library, è sufficiente creare una nuova pagina, dopo aver aggiunto un riferimento all'interno del sito web, così che l'assembly con la nostra classe per generare RTF possa essere referenziata ed utilizzata. Per prima cosa è dunque necessario estrarre i dati che saranno visualizzati. Per farlo si è scelto il database pubs di SQL Server, creando un report degli autori presenti, con questo codice:

```
// estrazione dei dati
SqlDataAdapter da = new SqlDataAdapter("SELECT
    au_lname as [Last Name], au_fname as [First Name],
                Address, City, ZIP, Phone FROM Authors",
 ConfigurationManager.ConnectionStrings["pubs"].Connect
                                          ionString);
DataTable dt = new DataTable();
da.Fill(dt);
```

Creata la DataTable e popolata con i dati, è necessario provvedere alla creazione del report vero e proprio. È utile sottolineare che la query è stata strutturata in maniera tale che i nomi dei campi vengano mostrati con un nome più descrittivo e migliore da visualizzare nel report stesso, utilizzando gli alias sulle colonne. A questo punto va creata l'istanza della classe **RtfWriter**, che contiene il nostro motore di generazione del report:

```
// Generazione del report
RtfWriter rtf = new RtfWriter("report.rtf");
```

E successivamente vanno aggiunti un po' di paragrafi, prima della tabella vera e propria, così:

```
// intestazione
rtf.WriteParagraphStart();
rtf.WriteBold("Report degli autori");
rtf.WriteParagraphEnd();
rtf.WriteEmptyParagraph();
rtf.WriteParagraph("Generato " +
        DateTime.Now.ToLongDateString() + " alle ore " +
                DateTime.Now.ToLongTimeString(), 0);
rtf.WriteEmptyParagraph();
```

Infine, non resta che passare la DataTable al metodo WriteTableFromDataTable, che genererà il codice RTF, attraverso il codice che si è analizzato nei paragrafi precedenti:

```
// converto la DataTable in RTF
rtf.WriteTableFromDataTable(dt);
```

L'ultimo passaggio consiste nello scrivere a video il risultato, dopo aver impostato l'header Content-Disposition, in modo che proponga al client di scaricare il file generato.

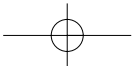
```
// scrivo a video
Response.Clear();
Response.ContentType="text/rft";
Response.AppendHeader("Content-Disposition",
        "attachment; filename=" + rtf.DocumentName);
Response.Write(rtf.GenerateDocument());
Response.End();
```

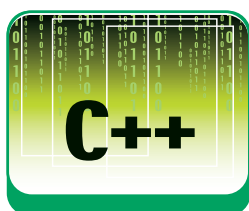
CONCLUSIONI

Lungi dall'essere completa al 100%, questa classe ha però diversi vantaggi e rappresenta una valida base di partenza per la creazione di moduli di reportistica anche più complessi di quello oggetto dell'esempio.
Il vantaggio principale che offre è senza dubbio quello di consentire una facile creazione di tabelle, oltre alla possibilità di essere utilizzato anche in altri ambiti, come ad esempio in presenza di Windows Service o WinForms, dato che la parte di generazione, che nel nostro esempio abbiamo utilizzato in una pagina web, può essere tranquillamente inserita in altri contesti.

Daniele Bochicchio

Tag	Descrizione
\par	Specifica la fine del paragrafo.
\pard	Specifica la fine del paragrafo e di qualsiasi altro stile specificato.
\line	Va a capo senza interrompere il paragrafo corrente.
\ql	Allinea il testo a sinistra.
\qr	Allinea il testo a destra.
\qj	Allinea il testo giustificandolo.
\qc	Allinea il testo al centro.
\b	Imposta il grassetto per il testo specificato.
\i	Imposta l'italico per il testo specificato.
\ul	Imposta la sottolineatura del testo.
\super	Inserisce il testo come apice.
\fin	Imposta il rientro della prima linea del paragrafo.
\lin	Imposta il margine del rientro a sinistra del paragrafo.
\rin	Imposta il margine del rientro a destra del paragrafo.
\page	Inserisce un salto pagina.
\paperwn	In n va specificato la larghezza della pagina.
\paperhn	In n va specificata l'altezza della pagina.
\margln	In n va specificato il margine sinistro della pagina.
\margrn	In n va specificato il margine destro della pagina.
\margtn	In n va specificato il margine superiore della pagina.
\margbn	In n va specificato il margine inferiore della pagina.
\landscape	Imposta la pagina come orizzontale.
\portrait	Imposta la pagina come verticale.

Tabella: I principali tag di RTF

TENERE LA MEMORIA SOTTO CONTROLLO

CONCLUDIAMO IL VIAGGIO NEL PAESE DEGLI SMART POINTERS, ANALIZZANDO QUELLI PIÙ AVANZATI OFFERTI DALLA LIBRERIA BOOST. LA CONOSCENZA DI QUESTI ELEMENTI È UN OBBLIGO PER OGNI PROGRAMMATORE C++

Benvenuti al secondo appuntamento col mondo degli smart pointers. Chi di voi si fosse perso il primo (male!), dovrebbe seriamente pensare di rimettersi in carreggiata provando a seguire questo. "Perché è così fondamentale?" direte voi "io ho programmato per anni senza sapere cosa significhi la parola smart pointer, e non ho mai avuto problemi!". Davvero?
Non vi è mai successo che la vostra applicazione si mettesse improvvisamente, inesorabilmente a ingurgitare risorse, fino a rallentare ed eventualmente mandare in crisi il sistema?
Non vi è mai successo che nel bel mezzo del programma comparisse ai vostri clienti una simpatica schermata di dump reporting, in seguito ad un errore di segmentation fault, o access violation?
Non avete mai pensato che le eccezioni causino più problemi di quanti non ne risolvano, quando utilizzandole il programma va incontro a strani effetti collaterali?
Tutti questi problemi hanno una sola causa: state continuando a gestire la memoria dinamica manualmente.
Quando dichiarate un oggetto, come in:

```
Musica canzone;
```

non dovete preoccuparvi di rimuoverlo, perché sapete che ci penserà l'applicazione al momento dell'uscita dal blocco attuale (per dirla meglio: dall'area di visibilità della variabile). Ma ogni volta che utilizzate una chiamata all'operatore *new*, avete l'obbligo di richiamare, prima della fine dell'applicazione, la corrispettiva istruzione *delete*. E qui cominciano i problemi:

- Se vi dimenticate di deallocare un oggetto creato con new, causate un **memory Leak**: il vostro oggetto rimane in memoria sotto forma di inutile spazzatura, occupando spazio inutilmente.
- D'altro canto, se vi sbarazzate troppo presto della risorsa (ciò che in gergo viene detto **premature free**) lasciate dei puntatori in giro per la vostra applicazione che fanno ancora riferimento ad un'area già distrutta. Questi puntatori prendono il nome di **dangling pointers**, e sono una delle cause principali del mal di testa cronico che affligge il programmatore C++ medio.
- Se accedete ad un'area di memoria scorretta (perché non avete ancora inizializzato l'oggetto con new, o perché state usando un dangling pointer), potete causare ogni tipo di danno, dal **segmentation fault** in su.
- Se, poi, distruggete una risorsa già distrutta, richiamando due volte di seguito l'operatore delete sullo stesso puntatore, avete praticamente la certezza di aver come minimo mandato in crash l'applicazione. Potete vantarvi di aver compiuto con successo un **double free**.

La soluzione a questi problemi esiste, e non è quella di mettersi a spulciare con un debugger per ore il proprio codice, nella speranza di capire in quale delle miriadi di oggetti usati dalla vostra applicazione si sia generato un dangling pointer. La soluzione è: **fate gestire la memoria dinamica automaticamente alla macchina**. Ci sono diversi sistemi per conseguire questo risultato: nel precedente articolo abbiamo cominciato a fare la conoscenza con le fondamentali classi note come **smart pointer**. Si tratta di classi proxy, ovvero oggetti che incapsulano un puntatore, ne imitano il comportamento e richiamano l'operatore delete al momento della loro distruzione. In questo modo gli smart pointer possono essere allocati staticamente sullo stack, come l'oggetto Musica che abbiamo visto all'inizio, e contemporaneamente garantiscono la deallocazione automatica e sicura delle risorse che detengono in memoria dinamica. Un esempio vale più di mille parole:

```
#include <memory>
#include <iostream>
```

```
int main() {
    //creiamo la risorsa con new
    std::auto_ptr<string> stringa(new
                                string("Ciao mondo!");
    //... ma non la distruggiamo! Ci pensa
                                  l'applicazione.
} //qui l'auto_ptr "stringa" viene distrutto, e con lui
                              la risorsa "Ciao mondo".
```

Nel primo appuntamento di questa serie abbiamo cominciato a vedere due smart pointer fondamentali: boost::scoped_ptr e std::auto_ptr. Abbiamo cominciato a scoprire dei vantaggi fondamentali offerti da queste strutture, ma anche molte limitazioni, in particolare:

- Lo smart pointer *boost::scoped_ptr* è semplice e comprensibile, ma non è copiabile e pertanto non può uscire dall'area di visibilità in cui è stato dichiarato.
- Lo smart pointer *std::auto_ptr* può essere utilizzato per passare risorse da una funzione all'altra (attraverso un modello chiamato source/sink), ma ha uno strano sistema chiamato *copia distruttiva* che sembra creato apposta per rendere le cose complicate.

Quest'ultimo punto ha segnato la chiusura del nostro precedente appuntamento, ed è effettivamente importante e complesso. Nel prossimo paragrafo riprenderemo l'illustrazione del meccanismo di copia distruttiva, delle limitazioni di auto_ptr, e vedremo come superarle.

PROBLEMI DI CONDIVISIONE

L'amara verità alla quale il programmatore C++ deve presto rassegnarsi, è che l'auto_ptr fornito dalla libreria standard non può essere usato alla leggera, e in molti casi (qualcuno direbbe: in troppi casi) semplicemente non funziona. Il problema fondamentale è la ***copia distruttiva***, cioè il meccanismo per cui copiando un oggetto *A* in uno *B*, il possesso della risorsa passa a *B*, mentre A viene messo a terra. Esempio:

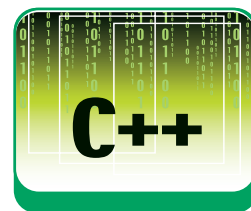
```
#include <iostream>
#include <string>
#include <memory>
using namespace std;
int main()
{
    //creiamo un nuovo auto_ptr<string>
                          chiamato "originale"
    auto_ptr<string> originale(new
                          string("Ciao, mondo!"));
```

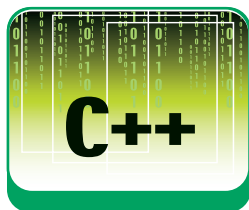
```
    //creiamo un auto_ptr<string> che copia
                                  l'originale
    auto_ptr<string> copia(originale);

    //ora "copia" possiede la stringa
    if (copia.get())
        cout << "La copia e' valida: " <<
                                *copia << endl;
    //ma "originale" è messo a terra!
    if (!originale.get())
        cout << "L'originale e' stato
                          messo a terra!\n";
}
```

Risultato:

```
La copia e' valida: Ciao mondo!
L'originale e' stato messo a terra!
```

Per capire perché il comitato di standardizzazione ha scelto un sistema così strano per gestire auto_ptr, ci si potrebbe chiedere cosa accadrebbe se due copie identiche di auto_ptr si trovassero a condividere la stessa risorsa?. Ma al momento della loro distruzione il sistema andrebbe incontro a un double free.
La *copia distruttiva*, però, ha un grave problema di fondo: modifica la semantica standard del linguaggio per la quale ci si aspetterebbe che un dato e la sua copia siano identici.
Detto in altre parole: usando auto_ptr occorre sempre stare attenti che la copia che stiamo utilizzando non sia quella di partenza, ormai "messa a terra". Ciò ha almeno un'implicazione molto pesante: dal momento che i **contenitori standard** non danno alcuna importanza all'ordine con cui effettuano le copie, **non è possibile usarli per contenere oggetti auto_ptr**.
Vale la pena di notare che questi problemi affliggono non solo gli auto_ptr, ma anche le classi composte da uno o più membri auto_ptr. Quest'esempio chiarisce la situazione:

```
#include <iostream>
#include <vector>
#include <memory>
#include <string>
using namespace std;
struct Musica
{
    string titolo;
    Musica(const string& _titolo) : titolo(_titolo)
                                              {};
};
class LettoreStereo
{
    auto_ptr<Musica> musica;
public:
```

BOOST!

La libreria boost (www.boost.org) è un riferimento essenziale per i programmatori C++ più smaliziati, dal momento che fornisce molte funzionalità avanzate e colma alcune lacune del linguaggio.

```
    LettoreStereo(const string& titolo) :
        musica(auto_ptr<Musica>(new
                        Musica(titolo))) {}
    void Play() {
        if (musica->titolo.empty())
            cout << "Lettore
                            vuoto.";
        else
            cout << "Sto suonando:
                " << musica->titolo;
    }
};
int main()
{
    vector<LettoreStereo> jukeBox;
jukeBox.push_back(LettoreStereo("Bach.MP3"));
    jukeBox[0].Play();
}
```

Qui si riprende parzialmente una situazione illustrata nel primo articolo, nel quale *Musica* era una classe base, e *LettoreStereo* doveva usare un auto_ptr per preservare il polimorfismo. Compilare e/o eseguire questo programmino è interessante anche per valutare la qualità dei vostri strumenti di lavoro. Un compilatore aderente allo standard deve generare un errore, con un messaggio simile a questo:

```
Error in line: "
  jukeBox.push_back(LettoreStereo("Bach.MP3")); "
    No matching function call to
        "LettoreStereo(const LettoreStereo&)"
Candidates are: "LettoreStereo(LettoreStereo&)"
```

È il risultato di un trucco che ha usato il comitato di standardizzazione nello stendere i requisiti di una classe auto_ptr, stabilendo che **il costruttore per copia di auto_ptr debba essere dichiarato con un parametro *non-const***. Dato che gli algoritmi di inserimento nei contenitori funzionano con un valore const, gli auto_ptr non possono essere inseriti nei contenitori standard. E la stessa cosa vale anche per le classi come *LettoreStereo*, che hanno almeno un membro auto_ptr, e non definiscono esplicitamente un costruttore per copia (secondo le regole del C++, infatti, queste classi assumono implicitamente un costruttore per copia non-const).
Se, d'altro canto, utilizzassimo un compilatore non aderente allo standard in questa direttiva, il comportamento sarebbe indefinito: Visual C++ 6.0, ad esempio, compila senza il minimo warning restituendo (per questa volta) il messaggio corretto. Altri compilatori, soprattutto in seguito ad operazioni di sorting, stampano a video "Lettore vuoto", facendo capire che è sopravvissuta la copia sbagliata.

Fig. 1: ***Tre smart pointer a reference counting condividono la stessa risorsa***

REFERENCE COUNTING

Se gli smart pointer fossero persone, quelli che abbiamo visto finora non sarebbero tanto simpatici. Scoped_ptr è un avaro che non vuol dividere la sua risorsa con nessuno, e auto_ptr è un ladro pronto a rubarla ai suoi stessi simili. Questi comportamenti chiusi ed egoisti riflettono una consapevole scelta di progettazione, cioè quella di far funzionare queste classi senza che esse debbano "mettersi d'accordo" fra loro.
Quando più smart pointer devono condividere la stessa risorsa, però, la politica autarchica non regge più, e queste classi devono necessariamente trovare un modo per comunicare, o quantomeno per far sapere alle altre che esistono. I modi in cui si può perseguire questo fine sono diversi e hanno molte varianti: qui ci occuperemo di analizzare alcune tecniche di **reference counting**, rimandando ai testi in bibliografia per gli altri sistemi (il settimo capitolo del testo di Alexandrescu offre una panoramica molto vasta sull'argomento).
Come illustrato in **figura 1**, nel sistema di reference counting gli smart pointer utilizzano un contatore per sapere quanti loro simili stanno accedendo alla risorsa, in questo modo:

- All'acquisizione della risorsa (tramite costruttore, *o reset*), viene creato un contatore.
- Alla copia (tramite costruttore per copia o assegnamento) il contatore viene incrementato.
- Alla distruzione il contatore viene decrementato, e se questo arriva a zero la risorsa viene distrutta.

Così facendo, si ottiene uno smart pointer "universale" che funziona senza problemi nella maggior parte dei casi (le eccezioni esistono, e le vedremo verso la fine dell'articolo. Abbiate pazienza.).

● **Shared_ptr**
La libreria boost (sempre lei!) offre uno smart pointer a reference counting di grande versatilità e semplicità d'uso. La classe è tanto utile e ben progettata che è stata presa a modello per lo smart pointer omonimo nel TR1, ed è praticamente certo che sarà parte integrante dello standard della nuova versione del C++ (C++0x), con il nome di tr1::***shared_ptr*** (molti libri e guide usano già questa denominazione. Io preferisco *boost::shared_ptr*, dal momento che per molti compilatori il TR1 è ancora "qualcosa che si mangia").

● **Funzioni per il conteggio**
A questo punto dovrei mostrarvi l'interfaccia di boost::shared_ptr, ma sarebbe uno spreco di spazio, perché ricalca la struttura già presentata in scoped_ptr (togliendo il discorso su *non-copyable*, ovviamente), con l'aggiunta di un paio di funzioni dedicate al contatore.

- **long use_count() const**: restituisce l'indice del contatore.
- **bool unique() const**: restituisce true se lo smart_pointer è l'unico a detenere la risorsa (cioè se use_count() == 1, anche se è solitamente più rapida dell'accesso a use_count).

Ecco un esempio di queste funzioni in azione:

```
#include <iostream>
#include <string>
#include <boost/shared_ptr.hpp>
using namespace std;
using namespace boost;
struct Musica
{
        string titolo;
        Musica(const string& _titolo) : titolo(_titolo)
                                                    {};
        ~Musica() {cout << titolo << " e' stata
                                   distrutta!" << endl;}
};

int main()
{
        shared_ptr<Musica> musica(new
                                  Musica("Bach.MP3"));
                {//sotto-scope
                shared_ptr<Musica>
                                          copia(musica);
                cout << "Ci sono " <<
                                     musica.use_count()
                              << " riferimenti a " <<
                                   musica->titolo << endl;
        }//qui la copia viene distrutta
        if (musica.unique())
                cout << "E' rimasto un solo
                                riferimento!" << endl;
}
```

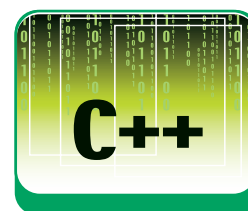
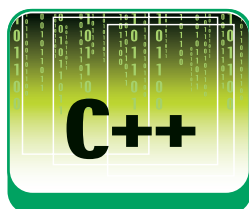
Risultato dell'esecuzione:

```
Ci sono 2 riferimenti a Bach.mp3
E' rimasto un solo riferimento!
Bach.Mp3 e' stata distrutta!
```

Va notato che il funzionamento dello smart pointer è completamente automatico, perciò non si è mai realmente interessati al numero o all'unicità dei riferimenti, se non per ragioni di pura curiosità o di debug (oppure se state creando un sistema *Copy On Write*, nel qual caso la funzione *unique()* è effettivamente utile).

● **Funzioni per il pointer_casting**
Funzioni più utili sono quelle relative al casting del puntatore, che è bene comprendere a fondo. Proviamo a ipotizzare di avere uno *shared_ptr<const Musica>*, e di volere invece ottenere a tutti i costi uno *shared_ptr<Musica>*, da passare ad una funzione.

```
void RinominaMusica(const shared_ptr<Musica>&
                  musica, const string& nuovoTitolo) {
        musica->titolo = nuovoTitolo;
}
int main()
{
        shared_ptr<const Musica> musica(new
                                  Musica("Bach.MP3"));
        //nota: questa "soluzione" è sbagliata!
        shared_ptr<Musica>
 musicaNonConst(const_cast<Musica*>(musica.get())
                                                    );
        RinominaMusica(musicaNonConst,
                                  "Beethoven.MP3");
        cout << musica.use_count() << endl; //
                                                  1!!!
}
```

Nel codice qui sopra ho provato a costruire un nuovo ***shared_ptr<Musica>*** prendendo il puntatore di musica tramite una chiamata al metodo *get()*, e sbarazzandomi del suo vincolo di costanza per mezzo di un *const_cast*. La cosa sembra aver senso, e compila pure, ma va in crash alla fine dell'esecuzione. Si può intuirne la

USARE BOOST
La libreria boost non è famosa per la semplicità d'installazione. Fortunatamente, le definizioni degli smart pointers non richiedono alcuna installazione: è sufficiente impostare l'IDE (o le variabili d'ambiente) inserendo fra le directory di inclusione la directory radice in cui si è scaricato boost.

ragione, osservando che la chiamata *musica.use_count()* restituisce 1: essendo due oggetti costruiti separatamente, *musicaNonConst* e musica hanno due contatori distinti, e quindi entrambi i distruttori deallocano la risorsa, provocando così un double free. Per superare questo genere di problemi, la libreria boost fornisce una funzione apposita, chiamata **const_pointer_cast**:

```
int main()
{
        shared_ptr<const Musica> musica(new
                                  Musica("Bach.MP3"));
        shared_ptr<Musica>
musicaNonConst(const_pointer_cast<Musica>(musica
                                              ));
        RinominaMusica(musicaNonConst,
                                  "Beethoven.MP3");
        cout << musica.use_count() << endl; // 2
                                        (giusto!)
}
```

Questa soluzione è corretta, come testimonia la chiamata a *musica.use_count()*, che restituisce giustamente 2 (un riferimento da musica, ed uno da *musicaNonConst*). *Const_pointer_cast* serve quindi ad ottenere uno smart pointer privo del vincolo di costanza, **che condivide il possesso con lo smart pointer di partenza**. Allo stesso modo vengono fornite le funzioni **static_pointer_cast, e dynamic_pointer_cast**, per emulare il comportamento degli operatori *static_cast* e *dynamic_cast.*

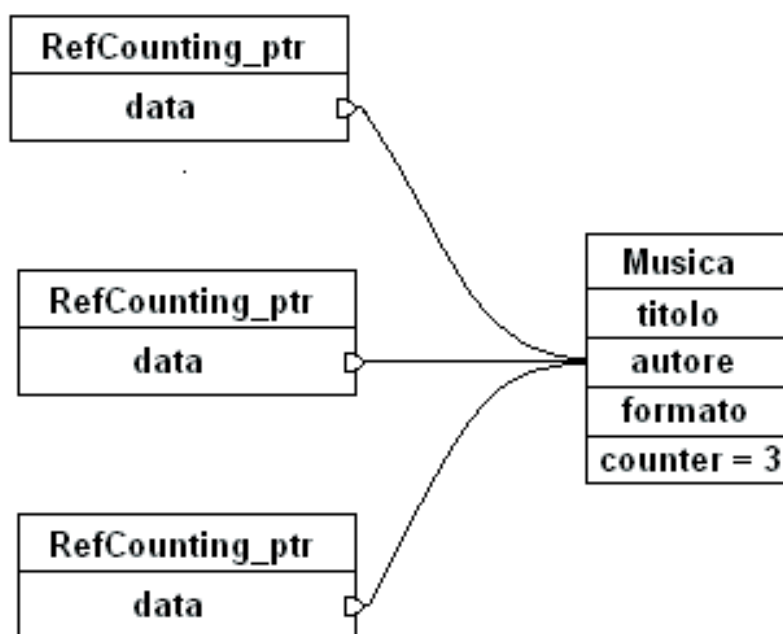



Fig. 2: ***Una tipica implementazione di smart pointer intrusivo, con il contatore incapsulato dall'oggetto puntato.***

● **Intrusive_ptr**

Se sono riuscito nel mio intento affabulatorio, a questo punto vi starete convertendo alle grazie di shared_ptr, e promettendo solennemente a voi stessi che mai più userete un auto_ptr in vita vostra. Dato che so di non essere così convincente, più probabilmente vi starete chiedendo quali siano i vantaggi di un auto_ptr tali da valere la seccatura della copia distruttiva. Rispondo io. A parte il fatto che in rari casi la copia distruttiva è proprio il comportamento che volete ottenere, gli shared_ptr hanno un costo, sia in termini di memoria, sia di prestazioni.

Dalla **figura 1** potete intuire la ragione principale di quest'overhead: ad ogni smart pointer viene aggiunto un puntatore al contatore dei riferimenti. Sono solitamente 4 byte in più da memorizzare, e del lavoro in più per ogni copia e distruzione.

Intendiamoci, normalmente non si sente minimamente il peso di un puntatore in più, ma se si vogliono realizzare migliaia di riferimenti shared_ptr, l'overhead diventa statisticamente significativo. Come spiegavo qualche paragrafo fa, la soluzione a questi problemi si raggiunge aumentando il "dialogo" fra lo smart pointer e il resto dell'applicazione: in questi casi, infatti, si può usare uno smart pointer *intrusivo* (v. **figura 2**).

Questo genere di smart pointer non ha bisogno di memorizzare la posizione del contatore, perché sa già dove si trova: all'interno dell'oggetto puntato. Così facendo ogni smart pointer intrusivo prende solo lo spazio di un puntatore, proprio come un auto_ptr. C'è un "però" piuttosto ovvio: l'oggetto puntato deve essere a conoscenza del fatto che riceverà le attenzioni degli smart pointer intrusivi, e deve quindi memorizzare il contatore e fornirne un'interfaccia di accesso.

In termini pratici, la libreria boost fornisce un ***intrusive_ptr*** come smart pointer intrusivo, e impone di sovraccaricare esternamente ad ogni classe da referenziare, le funzioni ***void intrusive_ptr_add_ref(Classe*)***, e ***void intrusive_ptr_release(Classe*)***, rispettivamente per aggiungere o togliere un riferimento. L'implementazione più ingenua e minimalista possibile di una classe che soddisfi questi requisiti è quella che segue:

```
struct IntrusiveCounted
{
        int counter;
        IntrusiveCounted() : counter(0) {}
};
inline void intrusive_ptr_add_ref(IntrusiveCounted*
                                              p)
{
    p->counter++;
```

```
}
inline void intrusive_ptr_release(IntrusiveCounted* p)
{
  if (!(--p->counter))
    delete p;
}
```

Si tratta davvero di un'implementazione naïf: il contatore è lasciato al pubblico scempio, le funzioni non sono incluse in alcun namespace, e la parola *thread-safety* è lontana anni luce! Tuttavia ho eliminato ogni orpello di design, per permettervi di valutare in modo immediato come funzioni una classe che debba essere referenziata da un *intrusive_ptr*. Poiché dovrete ripetere ogni volta tutte queste dichiarazioni, finirete col comportarvi come fanno tutti gli altri programmatori: scriverete una volta sola una bella implementazione completa di IntrusiveCounted, la salverete in un file tipo "IntrusiveCounted.hpp", e la userete come base da cui derivare pubblicamente.

```
[...] //altri include standard
#include <boost/intrusive_ptr.hpp>
#include <IntrusiveCounted.hpp>
//ora è sufficiente derivare pubblicamente da
                                    IntrusiveCounted
struct Musica : IntrusiveCounted
{
      string titolo;
      Musica(const string& _titolo) : titolo(_titolo)
                                                {};
      ~Musica() {cout << titolo << " e' stata
                              distrutta!" << endl;}
};
int main()
{
      intrusive_ptr<Musica> musica(new
                              Musica("Bach.MP3"));
      // [...] usate musica come uno shared_ptr
                                               [...]
}
```

Come vedete, una volta scritta la classe base, costruire un nuovo tipo che sia referenziabile da un intrusive_ptr, o aggiornare una classe preesistente, è davvero semplice e intuitivo. Più difficile è stabilire a priori quando è preferibile usare uno shared_ptr, e quando, invece, conviene un intrusive_ptr. In linea teorica bisognerebbe ricorrere ai primi quando si hanno più riferimenti che oggetti e ai secondi in caso contrario, ma la questione non è così netta. La linea consigliata dagli stessi creatori della libreria è: *se siete in dubbio, provate prima con shared_ptr.*

● Weak_ptr

All'inizio di quest'articolo vi ho detto che, se aveste avuto pazienza, vi avrei svelato uno dei casi in cui è possibile mandare in crisi perfino il robustissimo *shared_ptr*. La vostra iniziazione è finita, e ora possiamo parlarne. Proviamo a definire il comportamento di un eroe dell'antica Grecia, come mostrato nella **figura 3**:

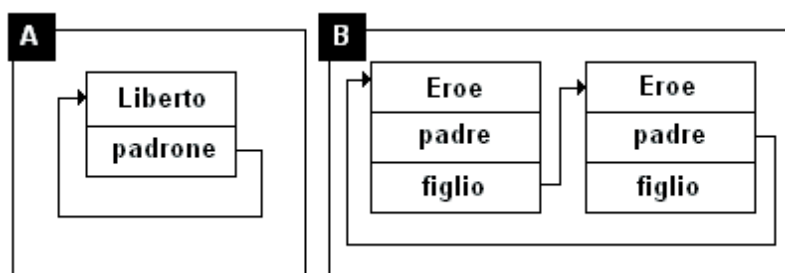
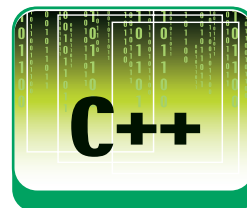



Fig. 3: ***Due riferimenti circolari in cui l'uso di shared_ptr causa un memory-leak. In A, un liberto diventa il padrone di se stesso; in B, padre e figlio si "sorreggono a vicenda".***

```
struct Eroe
{
      string nome;
      shared_ptr<Eroe> padre;
      shared_ptr<Eroe> figlio;
      Eroe(const string& _nome) : nome(_nome)
                                              {};
      ~Eroe() {cout << "Sventura! Il grande "
                  << nome << " e' morto!";};
      void Presentati() {
            cout << "Io sono " << nome;
            if (padre)
                  cout << ", figlio di " <<
                                    padre->nome;
            if (figlio)
                  cout << ", padre di "
                                << figlio->nome;
            cout << ".\n";
      }
};

int main()
{

      shared_ptr<Eroe> priamo(new
                               Eroe("Priamo"));
      shared_ptr<Eroe> ettore(new
                               Eroe("Ettore"));
      priamo->figlio = ettore;
      ettore->padre = priamo;
      ettore->Presentati();
      priamo->Presentati();
}
```

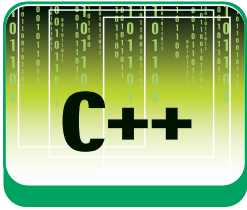
L'esecuzione di questo programmino ci dà l'impressione di aver fatto proprio un buon lavoro:

```
Io sono Ettore, figlio di Priamo.
Io sono Priamo, padre di Ettore.
```

SCOTT MEYERS
Effective C++ (la terza edizione ha degli ottimi riferimenti su shared_ptr)
More Effective C++ (ha una lunga sezione sull'implementazione "artigianale" di vari sistemi di reference counting)

HERB SUTTER
Exceptional C++ (numerosi riferimenti su auto_ptr e l'exception safety)
More Exceptional C++ (idem)

ANDREI ALEXANDRESCU
Modern C++ Design (presenta la classe smart_ptr di Loki)

BJÖRN KARLSSON
Beyond the C++ Standard Library (ottima introduzione a boost)

Ma… un momento! Dove sono i messaggi di "sventura" che dovrebbero essere stampati alla distruzione dei due eroi? Non ci sono, e ciò significa una cosa sola: **memory leak**. Il fatto è che lo shared_ptr *priamo->figlio* tiene in vita Ettore, e *ettore->padre* tiene in vita *Priamo*, tanto è vero che alla fine di main l'use count di entrambi è 2. Quando main termina, i due shared_ptr locali *priamo* ed *ettore* vengono distrutti, ma non le loro risorse, dal momento che lo use_count dei loro riferimenti interni è ancora 1! Come mostra questo esempio, i puntatori come shared_ptr tengono in vita l'oggetto al quale puntano, e vengono per questo definiti **puntatori forti**. Per spezzare il ciclo, occorre usare un **puntatore debole**. Finora conosciamo un solo puntatore debole: *Eroe** - ovverosia il *puntatore stupido*. Ma usarlo non è una buona idea.

```
//in questo esempio, Eroe::Padre è un puntatore
                                          stupido
int main()
{
        shared_ptr<Eroe> priamo(new
                                Eroe("Priamo"));
        shared_ptr<Eroe> ettore(new
                                Eroe("Ettore"));
        priamo->figlio = ettore;
        ettore->padre = priamo.get();
        priamo.reset(); //uccidiamo Priamo
        ettore->Presentati();
}
```

L'esecuzione di questo programma va in crash, con un output del genere:

```
Sventura! Il grande Priamo e' morto!
Io sono Ettore, figlio di (
```

Stavolta Priamo muore, secondo natura, ma ciò causa un dangling pointer nel membro padre di Ettore, ormai rimasto orfano, che "si commuove" al momento di dichiarare i suoi natali, mandando in crash l'applicazione.

Per risolvere il problema correttamente, ci vuole uno smart pointer **debole**, che non tenga in vita l'oggetto puntato, e boost mette a disposizione anche questo: **weak_ptr**.

Ci sono un paio di cose importanti da sapere su weak_ptr: la prima è che è possibile usare il metodo expired() per sapere se il riferimento al quale punta è ancora in vita. La seconda è che weak_ptr non permette l'accesso all'oggetto puntato direttamente, ma espone il metodo lock(), che crea uno shared_ptr, che può a sua volta essere usato per utilizzare l'oggetto. In questo modo si ha una maggior sicurezza: se l'oggetto non esiste più, lo shared_ptr restituito sarà vuoto. Dangling pointer, addio! La definizione di Eroe deve cambiare così:

```
struct Eroe
{
        //[...]
        weak_ptr<Eroe> padre;
        shared_ptr<Eroe> figlio;
        //[...]
        void Presentati() {
                cout << "Io sono " << nome;
                if (!padre.expired())
                        cout << ", figlio di " <<
                            padre.lock()->nome;
                if (figlio)
                        cout << ", padre di "
                                << figlio->nome;
                cout << ".\n";
        }
};
```

Ora l'esecuzione del programma è finalmente corretta e priva di memory_leak:

```
Sventura! Il grande Priamo e' morto!
Io sono Ettore.
Sventura! Il grande Ettore e' morto! (
```

CONCLUSIONI

Finalmente potete riporre nella vostra cassetta degli attrezzi una serie di utensili universali, pronti a risolvervi molti problemi insidiosi. Per ora, il nostro viaggio nel paese degli smart pointer si conclude qui - ci aspettano altre avventure in altri luoghi affascinanti della Memoria, come il regno degli *allocatori* e quello dei *Garbage Collector*. Ma non abbiamo certo esplorato ogni loro possibilità, e vi prometto che torneremo presto ad approfondire questi argomenti. Gli shared_ptr, in particolare, permettono di usare dei ***custom deleter*** che valgono tutte le vostre attenzioni, e saranno sicuramente trattati fra queste stesse pagine in futuro. Non mancate!

Roberto Allegra

L'AUTORE

Per ogni richiesta/critica/suggerimento l'autore può (e deve!) essere contattato all'indirizzo articoli@robertoallegra.it

LOKI E SMART_PTR

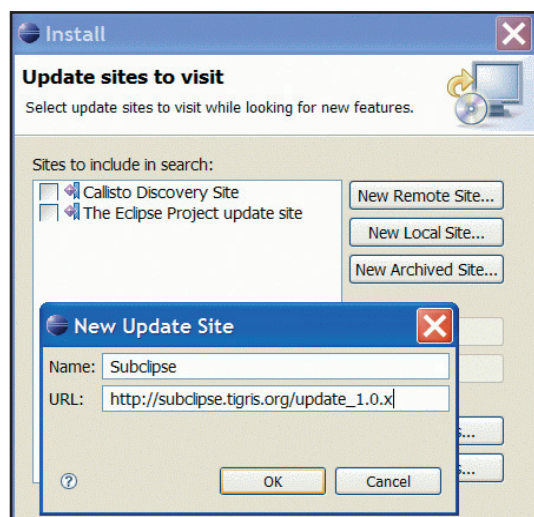
Lo smart pointer smart_ptr della libreria Loki è sicuramente uno dei risultati più interessanti conseguiti dal policy based design predicato da Alexandrescu. Definendo opportunamente i parametri del suo template, è possibile utilizzare smart_ptr come un puntatore forte, debole, intrusivo, a reference counting, reference linking, copy on write, e tante altre combinazioni. Ne riparleremo sicuramente su queste pagine, quando discuteremo questo rivoluzionario sistema di programmazione.

ECLIPSE NEL MONDO REALE

CHE USIATE ECLIPSE PER SVILUPPARE PROGETTI OPEN SOURCE O PER IL VOSTRO LAVORO IN AZIENDA, LE SUE INTEGRAZIONI CON I TOOL PIÙ USATI DEL MONDO JAVA POSSONO FACILITARVI LA VITA. VEDIAMO QUALI SONO GLI STRUMENTI A DISPOSIZIONE

Java è ormai un signore di mezza età, e il suo placido mondo è diventato una grande famiglia di standard di fatto: utility, librerie e framework. A volte sembra che non sia rimasto niente da inventare: se devo automatizzare la costruzione e l'installazione del mio sistema, userò probabilmente Ant o Maven; se devo testare il codice, la voce popolare mi suggerisce di usare JUnit; e così via. E visto che la "I" della parola "IDE" sta per "integrato", mi aspetto che il mio sistema di sviluppo sia buon amico di tutti questi arnesi, e mi permetta di usarli con la stessa facilità con cui uso il suo editor o il compilatore Java.
Per fortuna Eclipse è uno tra i membri più socievoli della famiglia Java. Grazie alla sua struttura aperta possiamo scaricare rapidamente nuovi componenti per integrarlo con gli altri strumenti di sviluppo. E spesso non ne abbiamo nemmeno bisogno, perché la distribuzione standard di Eclipse è già integrata con quasi tutti i tool davvero indispensabili. In questo articolo parleremo dell'integrazione tra Eclipse e tre utensili importanti per la vostra cassetta degli attrezzi Java: Subversion, JUnit e Ant.

PRIMA PARTE ECLIPSE SU MISURA

Per prima cosa vediamo come scaricare e installare un plug-in che integra Eclipse con uno strumento esterno. Cominciamo con il sistema di versionamento, che descriveremo in dettaglio nella seconda parte dell'articolo.
La distribuzione standard di Eclipse supporta già CVS, il più popolare version manager open source. Se selezionate il comando di menu *Window->Open Perspective->Other*, vedrete che avete già una "prospettiva CVS". Ma in questo articolo useremo un sistema di versionamento un po' più moderno: Subversion.
Subversion (a volte abbreviato in SVN) è il candidato per la successione al trono di CVS. E' probabile che prima o poi Eclipse decida di supportare Subversion nella distribuzione standard, ma fino a quel momento dobbiamo installare un plug-in apposta. Basta una rapida ricerca con Google per individuare Subclipse, un plugin sviluppato dagli stessi autori di Subversion. Il plugin contiene tutto quello che vi serve per connettervi ad un repository Subversion da Eclipse. Connettetevi a Internet e scegliete *Help->Software Updates->Find* and Install. Apparirà una finestra che vi chiede se vole-

SOTTO VERSIONE

C'è chi dice che per scrivere software servono tre cose: un computer, un compilatore e un sistema di versionamento. I primi due strumenti sono riconosciuti da tutti come indispensabili, ma non tutti conoscono l'importanza del terzo. Diciamo che potete fare a meno del versionamento, a patto che adoriate passare ore per cercare di capire chi ha modificato un file; che vi piaccia l'idea di perdere l'unica versione funzionante del vostro progetto; che i problemi di integrazione siano per voi un piacere sublime; e che non abbiate niente in contrario a cercare file di sorgente tra migliaia di vecchi messaggi di mail. In caso contrario, vi conviene usare il versionamento. I sistemi di versionamento sono bestiole complicate, e in questo articolo non abbiamo lo spazio per parlarne a lungo. Potete leggere i concetti di base nel box "Io e il repository". Vi incoraggiamo a studiare il resto per conto vostro. Non avete scuse: i "version control system" più popolari, come CVS e Subversion, sono belli gratis.

te cercare aggiornamenti per i componenti già installati, o installarne di nuovi. Selezionate la seconda opzione (*Search for new features to install*) e premete Next. La finestra successiva contiene la lista dei siti di update. Aggiungiamo il sito di Subclipse alla lista: scegliete *New remote site*, inserite l'indirizzo *http://subclipse. tigris.org/update_1.0.x* e dategli un nome qualsiasi.

Selezionate il nuovo sito di update e cliccate su *Finish*. Eclipse contatterà il sito di update e vi mostrerà quello che ci ha trovato sopra. Mentre scriviamo, l'unico plugin disponibile su questo sito è la versione 1.0.3 di Subclipse. Selezionatelo e cliccate su *Next* per proseguire con l'installazione. Le opzioni successive sono semplici: dovete accettare la licenza, decidere dove mettere i file del plugin (per default finisce nella stessa directory dei plugin standard di Eclipse) e confermare l'installazione. Quando avrete finito, Eclipse vi chiederà di riavviare il workbench.

Per verificare che l'installazione abbia avuto successo, scegliete Help->About Eclipse SDK e cliccate su *Feature Details*. Dovreste vedere tra le altre una feature di nome Subclipse. Verificate che funzioni: scegliete *Window->Open Perspective->Other* e controllate che nella lista delle prospettive ce ne sia una che si chiama *SVN Repository Exploring*. Apritela e andiamo avanti.

SECONDA PARTE IL CODICE

Ora che abbiamo installato Subclipse, possiamo usarlo per accedere ad un progetto conservato in un repository Subversion. Se non avete esperienza di versionamento, leggete i box "Sotto versione" e "Io e il repository".

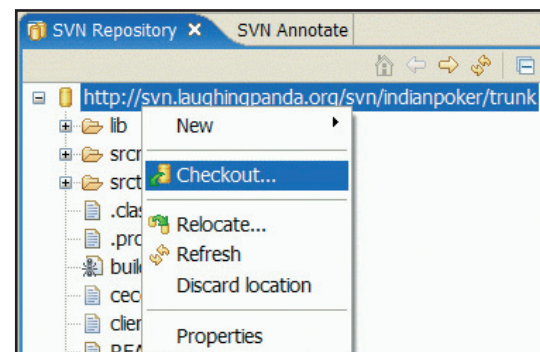
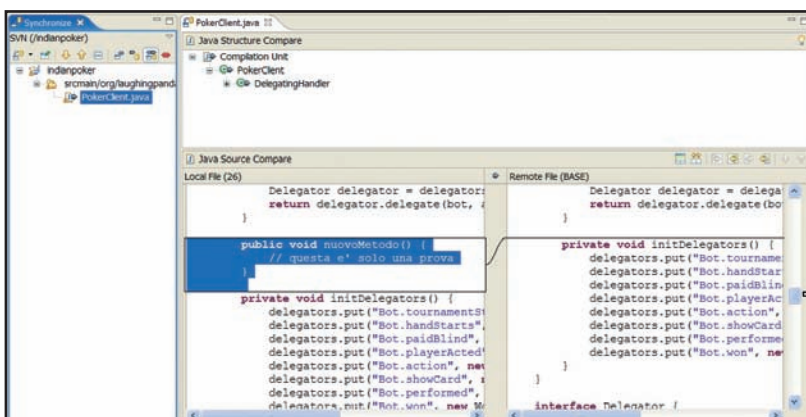
Se avete seguito il paragrafo precedente, dovreste essere di fronte alla prospettiva SVN Repository Exploring. Cliccate con il tasto di destra in questa vista e scegliete *New->Repository* Location. Come URL, inserite http://svn.laughingpanda.org/svn/indianpoker/trunk/ e premete Finish. Nella vista sulla sinistra apparirà un nuovo repository, che potete esplorare come se fosse un albero di folder locali. Tra gli altri file, ne vedrete uno che si chiama *.project* – questo è un indizio del fatto che il repository contiene un progetto Eclipse. Cliccate sul repository con il tasto di destra e scegliete *Checkout* per scaricare il progetto in locale.

Selezionate tutte le opzioni di default nelle finestre che seguono. Abbiate pazienza, ci vorrà un po' per scaricare tutti i file del progetto. Alla fine tornate alla prospettiva Java. Dovreste vedere nel *Package Explorer* un progetto di nome *indianpoker* (vedi box: "Poker all'indiana").

Ora che avete scaricato il progetto potete modificarne i file, ma non potete inserire le vostre modifiche nel repository – per farlo dovreste entrare nel team di sviluppo e farvi dare una password. Però potete farvi un'idea di come si lavora con uno strumento come Subversion. Modificate un qualsiasi file del progetto e salvatelo (Subclipse dovrebbe identificare i file e i folder modificati con un piccolo asterisco). Cliccate con il tasto di destra sul file modificato e selezionate il menu *Team*. Questo menu contiene i comandi del sistema di versionamento. Ad esempio: il comando *Commit* permette ai membri del team di inserire le proprie modifiche nel repository, *Update* permette di scaricare le modifiche di qualcun altro, e *Revert* cancella le modifiche locali e riporta il file allo stato in cui era durante l'ultimo update. Scegliete il comando *Synchronize with Repository* per aprire la prospettiva *Team Synchronizing*. In questa prospettiva potete confrontare la versione locale del progetto con quella nel repository. Ad esempio potete vedere cosa avete cambiato voi e cosa hanno cambiato gli altri. La vista *Synchronize* mostra le differenze tra i file locali e il repository. Cliccando sulle icone in cima alla vista potete vedere i file modificati localmente, quelli modificati nel repository, entrambi, o i file "in conflitto" (quelli che sono stati modificati sia localmente che nel repository).

Cliccate con il tasto destro sul file che avete modificato e scegliete *Open In Compare Editor*. Si aprirà un editor che mostra le differenze tra la versione locale e quella remota del file.

Un'altra funzionalità utile del sistema di versiona-

mento è quella che mostra la storia passata di ciascun file. Tornate alla prospettiva Java e selezionate una classe qualsiasi – ad esempio *org.laughingpanda.games.poker.indian.domain.Card*. Cliccateci sopra col tasto destro e scegliete *Team->Show* in *Resource History*. Apparirà una vista di nome *SVN Resource History* che mostra tutte le precedenti versioni del file, ciascuna con la sua data, il nome dell'autore ed il commento inserito dall'autore durante il commit. Potete anche selezionare due versioni per esplorarne le differenze. Con uno strumento così è facile mantenere il controllo su quello che succede al codice.

TERZA PARTE LA RETE DI SICUREZZA

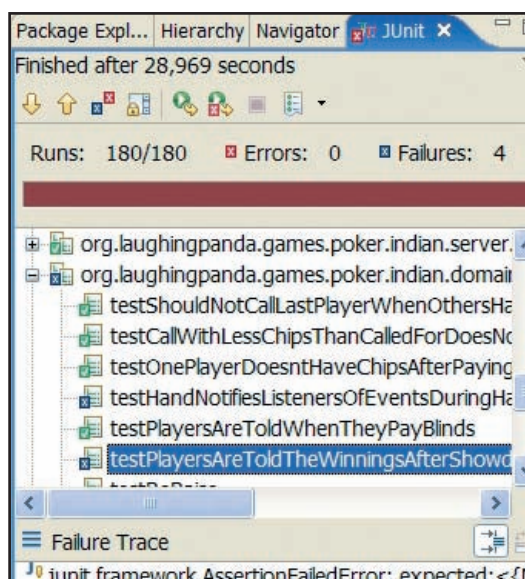
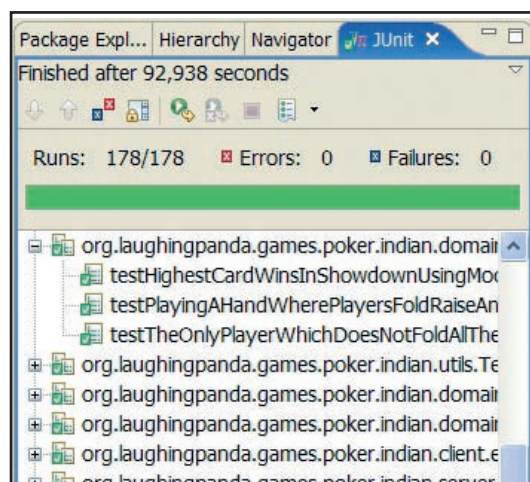
Diamo un'occhiata all'integrazione tra Eclipse e il framework JUnit (leggete il box "Tanti piccoli test" se non conoscete JUnit). Il progetto che stiamo usando come esempio contiene due folder di sorgenti. Una (*srcmain*) contiene il codice "di produzione", l'altra (*srctest*) contiene i test. La struttura dei package è la stessa per i due folder. Questo è uno dei tanti modi di organizzare i test, ma non l'unico. In certi progetti i test sono negli stessi folder del codice di produzione, o in package separati.
Esplorate il codice dei test per farvi un'idea di come funziona. In generale ciascun test è un metodo (un "test case") all'interno di una classe (il "test"). JUnit usa delle semplici convenzioni per trovare ed eseguire i test – ad esempio, le classi di test ereditano dalla classe *junit.framework.TestCase*, e ciascun metodo di test ha un nome che inizia per "test".
Cliccate con il tasto di destra sul progetto, e scegliete *Run As->JUnit Test*. Eclipse andrà a cercare tutti i test unitari nel progetto (nello stesso modo è possibile eseguire un singolo test, o tutti i test sotto un certo folder). Apparirà una vista *JUnit*. Man mano che i test vengono eseguiti vedrete avanzare una barra verde. Se anche uno solo dei quasi duecento

test del progetto test fallisce, la barra diventa rossa. Vediamo cosa succede se introduciamo un bug nel codice. Tornate al Package Explorer e guardate nel folder *srcmain*. Cercate la classe che rappresenta una "mano" del gioco: *org.laughingpanda.games.poker.indian.domain.Hand*. Questa classe contiene un metodo per verificare se tutti i giocatori hanno avuto modo di fare la loro puntata:

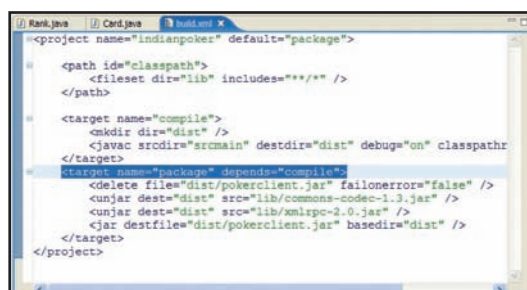
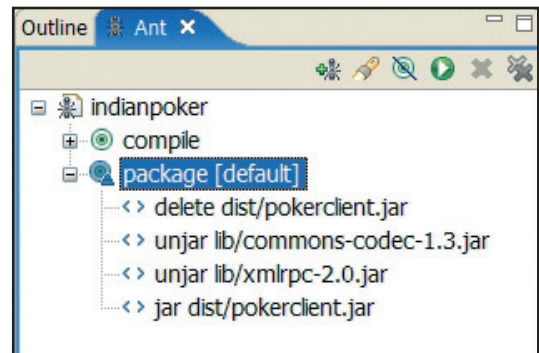
```
private boolean allPlayersHaveHadChanceToAct(int
                                        actions) {
  return actions >= players.size();
}
```

Modificate il confronto per introdurre un subdolo bacherozzo:

```
private boolean allPlayersHaveHadChanceToAct(int
                                        actions) {
  return actions > players.size();
}
```

Ora fate girare nuovamente i test, e vedrete apparire una barra rossa. JUnit vi segnalerà anche quali test sono falliti. Se cliccate sui test falliti, vedrete uno "stacktrace" dell'errore che vi permette di arrivare direttamente al codice colpevole.
Grazie ai test unitari, abbiamo buone chance di identificare gli errori prima che causino danni. Provate a introdurre errori in diverse parti del codice per vedere quali sono ben testate.

QUARTA PARTE IL PICCOLO MURATORE

Prima di finire, parliamo dell'integrazione tra Eclipse e Ant, il più popolare sistema di build per Java.
Costruire un sistema Java è semplice solo se il sistema è semplice. In questo caso basta compilarlo e lanciarlo. Ma la tipica "build" del sistema richiede molte altre operazioni. Di solito voglio impacchettare le mie classi Java in una serie di file JAR, aggiungere tutte le librerie scaricate da Internet, far girare i test per controllare che tutto funzioni ancora, e magari anche fare il "deploy" del sistema, cioé installarlo e renderlo operativo su una macchina di test o direttamente in produzione. Alla fine, non è raro che per un piccolo aggiornamento (ad esempio, per correggere un bug) si perda più tempo per tutte queste operazioni che per lavorare sul codice. Capita di incontrare progetti nei quali mettere un sistema in produzione richiede un intero giorno di lavoro.
Per fortuna c'è Ant, il più famoso sistema di build per Java. Ant è una versione riveduta e corretta del vecchio programma make usato dagli sviluppatori C. Permette (con un po' di lavoro) di scrivere degli script che automatizzano le operazioni lunghe e noiose di build e deploy. Lo script è un file XML (di solito si chiama *build.xml*) che contiene una serie di compiti da eseguire (Ant li chiama "target"). In questo articolo non spieghiamo in dettaglio come funziona Ant, ma se lavorate in Java probabilmente lo avete già incontrato. Eclipse supporta già Ant senza bisogno di aggiungere alcun plug-in esterno.
Andate nella root del progetto *indianpoker* e troverete un file di nome *build.xml*. Apritelo nell'editor. L'editor Ant di Eclipse è in grado di fare cose complicate come trovare gli errori nello script (il che, come potete immaginare, è un bel vantaggio quando si lavora con un programma scritto in un dialetto di XML).
Questo particolare file contiene solo due target. Il target *compile* compila tutte le classi (ma non i test) con il compilatore *javac*, e il target *package* impacchetta i file del client (e le librerie necessarie) in un unico file *jar* sotto la directory *dist*. Il secondo target "dipende" dal primo – questo significa che se lanciate *package*, Ant garantisce che venga prima eseguito *compile*.

```
<project name="indianpoker" default="package">

    <path id="classpath">
        <fileset dir="lib" includes="**/*" />
    </path>

    <target name="compile">
        <mkdir dir="dist" />
        <javac srcdir="srcmain" destdir="dist" debug="on" classpathr
    </target>
    <target name="package" depends="compile">
        <delete file="dist/pokerclient.jar" failonerror="false" />
        <unjar dest="dist" src="lib/commons-codec-1.3.jar" />
        <unjar dest="dist" src="lib/xmlrpc-2.0.jar" />
        <jar destfile="dist/pokerclient.jar" basedir="dist" />
    </target>
</project>
```

Quando un file di build contiene molti target, capire la struttura del file leggendo l'XML diventa difficoltoso. Meglio aprire una vista apposta. Scegliete *Window->Show View->Ant* per mostrare la vista *Ant*. Trascinateci dentro il file *build.xml* per vedere i due target. Il target *package* è marcato con una freccina per indicare che è il target di default – quello che viene eseguito automaticamente se si lancia la build senza specificare quale target eseguire.

L'idea è che per "rilasciare" il progetto si lancia il target *package*, che costruisce la libreria necessaria per implementare i client del gioco. Poi si distribuisce questo file ai giocatori. Proviamoci. Andate sul file *build.xml* nel *Package Explorer* (o sul target *package* nella vista *Ant*) e selezionate Run *As->Ant Build* dal menu contestuale. Vedrete Ant sparare un po' di informazioni nella Console. Quando la build sarà terminata (dopo pochi secondi) cliccate col tasto destro sul progetto *indiapoker* e scegliete *refresh* per chiedere ad Eclipse di allineare il progetto al vostro file system. Vedrete nel progetto un nuovo folder *dist* che contiene, oltre a un po' di file temporanei, anche il file *pokerclient.jar*.

CONCLUSIONI

In questo articolo abbiamo visto, in piccolo, un ciclo di sviluppo completo: abbiamo estratto un progetto dal suo repository, lo abbiamo modificato, abbiamo fatto girare i test e abbiamo lanciato la build per ottenere i file da distribuire. Tutto questo è stato possibile grazie all'integrazione di Eclipse con una serie di tool esterni.
Ma forse nel vostro caso vi tocca integrare Eclipse con JBoss, o con Tomcat, o con Maven, o con uno qualsiasi degli altri strumenti tipici del mondo Java. Nessun problema: cercate su Internet, e probabilmente troverete un plug-in pronto per l'uso. Le famiglie numerose come quella di Java hanno i loro svantaggi, ma almeno si ha la sicurezza di non essere mai da soli.

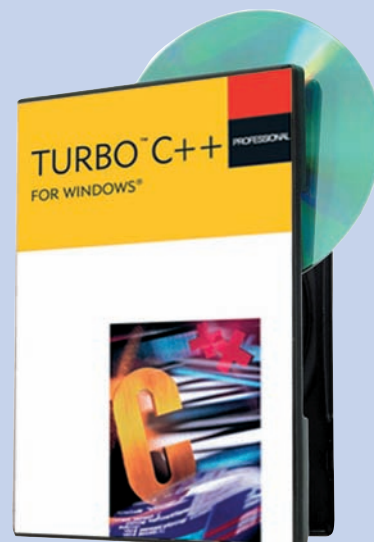
Paolo Perrotta

SOFTWARE SUL CD

Turbo C++ per Windows 2006

C++ SECONDO BORLAND

Borland è una delle software house storiche nel mercato dello sviluppo. Non si dimentica che proprio Borland ha sviluppato uno dei primi linguaggi completamente ad oggetti: l'object Pascal già a metà degli anni 90. Allo stesso modo è da attribuirsi a Borland anche l'avvento degli IDE Rad con il mitico Delphi. Per qualche anno si era allontanata dal mercato dei tool di sviluppo rivolti alla produttività individuale e solo di recente è ritornata con prepotenza ad occupare questo settore. La nuova linea di prodotti prende il nome di Turbo e vi appartengono Turbo C#, Turbo Delphi, e anche questo meraviglioso Turbo C++ per Windows. Compilatore ultraveloce dotato delle caratteristiche che da sempre hanno fatto grande i prodotti Borland. Prima di tutto una forte attitudine all'uso dei "componenti", ve ne sono circa 200 inclusi in questo pacchetto e poi un'IDE straordinariamente ricco, affidabile e stracolmo di features che facilitano la vita al programmatore. A fare da contorno a tutto questo un compilatore che produce codice altamente ottimizzato e veloce. Caratteristica essenziale per chi sceglie C++ come riferimento nei propri progetti di sviluppo

Directory:/turbocpp

JAVA SE DEVELOPMENT KIT 5.0 UPDATE 9

IL COMPILATORE INDISPENSABILE PER PROGRAMMARE IN JAVA

Se avete intenzione di iniziare a programmare in Java oppure siete già dei programmatori esperti avete bisogno sicuramente del compilatore e delle librerie Java indispensabili. Sotto il nome di Java SE Development Kit vanno appunto tutti gli strumenti e le librerie nonché le utility necessarie per programmare in JAVA. L'attuale versione è la 5.0 Update 9, ad un passo dall'attesissima versione 6 e immediatamente precedente al rilascio del codice di Java sotto forma OpenSource.

Directory:/j2se

ECLIPSE SDK 3.2.1

L'IDE TUTTOFARE

Eclipse è un progetto completo portato avanti da Eclipse Foundation con la collaborazione di una miriade di aziende fra cui IBM, Adobe, Sun e che si è prefissata lo scopo di creare un IDE estendibile per plugin adattabile a qualunque tipo di linguaggio o tecnologia. Di default Eclipse si propone come IDE per Java ed è qui che da il meglio di se. Ma proprio grazie ai suoi plugin è possibile utilizzarlo come ambiente di programmazione per PHP, per C++, per Flex e per molti altri linguaggi ancora. Inoltre sempre grazie per ciascun linguaggio sono disponibili altri plugin ad esempio per rendere l'ambiente RAD o per favorire lo sviluppo dei Web Services o altro. Insomma lo scopo è stato raggiunto completamente. Eclipse è realmente un IDE tuttofare, ormai maturo, e che serve una miriade di programmatori grazie alle sue caratteristiche di affidabilità e flessibilità. Unica nota negativa: una certa pesantezza che lo rende idoneo ad essere usato solo su PC con una dotazione hardware minima di tutto rispetto

Directory:/eclipse

PHP 5.2.0

IL LINGUAGGIO DI SCRIPTING PIÙ AMATO DEL WEB

Sono tre le colonne portanti di Internet: PHP, APACHE e MySQL. Certo la concorrenza è forte. Asp.NET e SQL Server avanzano con celerità, ma a tutt'oggi non si può affermare che i siti sviluppati in PHP costituiscano la stragrande maggioranza di Internet. Quali sono le ragioni del successo di cotanto linguaggio? Prima di tutto la completezza. PHP ha di base tutto quello che serve ad un buon programmatore, raramente è necessario ricorrere a librerie esterne, e quando è proprio indispensabile farlo esistono comunque una serie di repository che rendono tutto immediatamente disponibile ed in forma gratuita. Il secondo punto di forza del linguaggio sta nella sua capacità di poter essere utilizzato sia in modo procedurale che nella sua forma ad oggetti certamente più potente e completa. Esiste un terzo

di punta di forza essenziale che è quello riguardante la curva di apprendimento. PHP è in assoluto uno dei linguaggi con la curva di apprendimento più bassa nel panorama degli strumenti di programmazione. Si tratta perciò di uno strumento indispensabile per chi si avvicina alla programmazione web, a meno che non intendiate scegliere strade diverse quali possono essere ASP.NET o JSP

Directory:/php

PHALANGER 2.0

PHP IN TECNOLOGIA .NET

Ok, siete dei programmatori PHP ma vorreste che il vostro sito girasse in tecnologia .NET. Sapete che il miglio modo per far girare un sito sotto IIS sfruttandone tutte le potenzialità è scrivere codice .NET. Come fate? Facile! La soluzione esiste e si chiama Phalanger. Scrivete il vostro codice in PHP ma in realtà ottenete una pagina compilata in .NET. E' necessario installare qualcosa sul server, ma i risultati sono entusiasmanti!

Directory:/phalanger

NEMERLE 0.9.3

L'EMERGENTE SIMILE AL C#

Di tanto in tanto in programmazione arrivano delle novità assolute che inizialmente sfuggono al grande

pubblico per poi diffondersi lentamente nel corso del tempo ed arrivare alla piena maturità in un periodo di tempo abbastanza lungo. E' stato il caso di Ruby molti anni fa e di Python ancora prima. E'il caso di Boo e di Nemerle adesso. Nemerle è un linguaggio nato per la piattaforma .NET con una sintassi molto simile al C#. Si tratta di un linguaggio fortemente tipizzato ed orientato agli oggetti ma con la possibilità quando è il caso di essere utilizzato nella sua forma procedurale e da questo punto di vista segue i concetti cari al PHP. Altro aspetto interessante è quello di poter utilizzare le macro, mentre si avvicina per alcuni aspetti a Python quando si tratta di usare le liste ad esempio. Si tratta di un progetto interessante che merita una particolare attenzione. Nonostante la sua giovane età sembra infatti promettere alcune funzionalitò che potrebbero renderlo molto diffuso anche in breve tempo

Directory:/nemerle

BOO 0.7.6

SEMPLICE ED ELEGANTE

Mettiamo che vi piaccia Python ma non amate molto dovervi servire di librerie esterne, inoltre siete affascinati da .NET ma non vi piacciono i linguaggi che sottendono a questa tecnologia, fate uno + uno ed ecco il Boo. E' esattamente il ragionamento fatto da Rodrigo Barreto de Oliveira il quale aveva proprio voglia di un linguaggio simile al Python e che supportasse .NET e MONO ed ha tirato fuori il BOO. E nonostante il nome si presti a mille giochi di parole, almeno per la lingua italiana, bisogna dire che il linguaggio sta già ottenendo un discreto successo. Soprattutto Mono sembra già spingerlo a sufficienza e tuttavia non tarderemo a vederne gli effetti anche su piattaforma Windows

Directory:/boo

RUBY.NET BETA 5

IL NUOVO CHE AVANZA

Ruby, probabilmente lo conosce tutti. E' un linguaggio nato ormai da una decina d'anni inizialmente con la pretesa di essere uno strumento matematico dalle caratteristiche avanzate. Nel tempo si è evoluto e migliorato fino a raggiungere oggi una piena maturità che lo colloca fra i linguaggi più usati dell'ultimo anno. Proprio uno dei fra-

mework più noti sul web: Ruby On Rails ha vinto l'anno scorso il premio come miglior framework per lo sviluppo internet esistente, a dispetto di molti colosssi dell'informatica che operano in questo campo ed a testimonianza della grande maturità che questo linguaggio ha raggiunto. La sintassi è semplice, il linguaggio elegante, completo e particolarmete versatile, la curva di apprendimento molto bassa. Si tratta di uno strumento rapido che può coadiuvare lo sviluppo tradizionale e perché no? in molti casi sostituirlo. In questo numero di ioProgrammo ve lo presentiam nella sua versione per .NET, particolarmente ottimizzato per Windows.

Directory:/ruby

IRONPYTHON 1.0.1

UN'IMPLEMENTAZIONE DI PYTHON IN TECNOLOGIA .NET

In molti conoscono Python. Si tratta di un linguaggio agile, dinamico, elegante, ad oggetti e multipiattaforma. Per le sue caratteristiche di estrema flessibilità, per la sua facilità nella curva di apprendimento, per la sua leggerezza si tratta di un linguaggio estremamente diffuso su piattaforma Linux dove viene utilizzato per automatizzare gran parte delle impostazioni di sistema. Anche sul Web Python ha trovato una sua collocazione ben precisa, pare infatti che sia il linguaggio di scripting su cui si basa buona parte di Google. In ambiente Windows si sta diffondendo a macchia d'olio, tanto che oltre ad una versione standalone dell'interprete ne è appena nata anche una versione per .NET, appunto IronPython. Le caratteristiche sono identiche a quelle classiche di Python, ma il codice prima di essere eseguito viene convertito nel MIL di .NET con le conseguenti ottimizzazioni

Directory:/ironpython

ANT COLONY OPTIMIZATION

OSSERVARE IL COMPORTAMENTO "SOCIALE" DI ALCUNI ANIMALI E INSETTI PUÒ ESSERE UN UTILE INSEGNAMENTO PER RIPRODURLO CON UN MODELLO ALGORITMICO. SI TRATTA DI UNA NUOVA FRONTIERA DELLO SVILUPPO: LA SWARM INTELLIGENCE

Il comportamento di alcuni animali ed insetti ha da sempre suscitato interesse e a volte meraviglia nell'uomo. Spesso ci siamo chiesti come sia possibile che stormi di rondini riescano a seguire rotte di volo così precise o siamo rimasti affascinati dalle geometrie descritte da sciami di api. Studi sulla biologia di tali animali hanno dato molte risposte a riguardo. Ma tra tutti, l'insetto che suscita il maggiore interesse per il suo atteggiamento cooperativo è la formica. Negli ultimi anni nel ambito della programmazione, tali comportamenti hanno costituito le basi per la costruzione di nuovi metodi e modelli di risoluzione di problemi. Con la sigla SI (Swarm intelligence) ci si riferisce a questo nuovo approccio. In particolare Ant colony optimitation (ACO) introdotta nel 1992 dal nostro illustre connazionale Marco Dorigo si ispira al modo di agire di alcune specie di formiche. Vedremo come il comportamento sociale, quindi collettivo, sia stato simulato con sofisticati modelli algoritmici, e come tali modelli siano stati utilizzati per risolvere diversi categorie di problemi. Analizzeremo prima l'aspetto biologico da cui dedurremo il comportamento delle formiche, successivamente costruiremo un modello che lo descriva ed infine risolveremo un problema specifico con il metodo sviluppato.

LA BIOLOGIA DELLE FORMICHE

Quando le formiche hanno fame cercano cibo muovendosi casualmente. Nel momento in cui individuano del cibo, dopo essersi saziate, incominciano a portarlo verso il formicaio. A questo punto avviene un processo biologico significativo, incominciano a depositare scie di una sostanza chiamata feromone. Queste scie attraggono le altre formiche che così hanno la possibilità di individuare la fonte di cibo. Quando altre formiche mangiano e cominciano a tornare alla dimora i livelli di feromone crescono facendo aumentare l'attrattiva verso le zone in cui è presente il cibo. Dopo poco si creano vere e proprie colonne di formiche che trasportano gli alimenti dalla fonte al formicaio, chissà quante volte ci sarà capitato di osservarle. Va detto che il feromone per un processo di evaporazione con il passare del tempo diminuisce la sua azione fino ad esaurirsi. È quindi evidente che l'attrattiva verso una determinata zona è "significativa" se vi sono più formiche che hanno raggiunto la zona e che quindi nel tempo mantengono alti i livelli di feromone. Questa ultima precisazione sottolinea la condotta sociale delle formiche. In altri termini solo grazie all'azione collettiva si stabilisce un comportamento utile a tutta la colonia che ha come obiettivo la ricerca del cibo. Vi sono molti esperimenti riportati in testi di biologia che dimostrano tale condotta. Ad esempio, si è provato a dividere la sorgente di cibo dal formicaio con un ponte a due rami di lunghezza uguale (figura 1). Si è verificata una scelta casuale sul ramo da imboccare nel viaggio di andata alla ricerca di cibo. La scelta del ramo del ponte nel tragitto di ritorno, dal cibo alla dimora, influenzava le altre formiche che percorrevano nella maggior parte dei casi proprio quel ramo. Ciò è spiegato scientificamente proprio dal richiamo dovuto alle scie di feromome depositate dalla formica che ha trovato cibo.

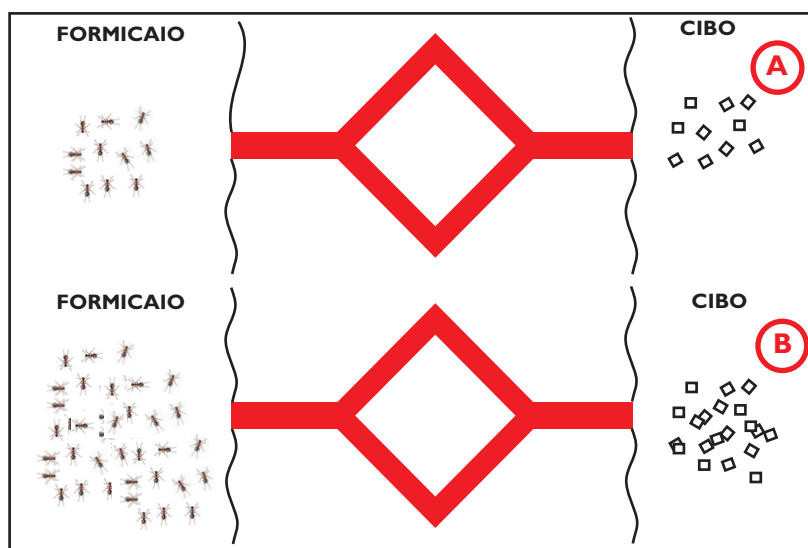



Fig. 1: *I due tipi di ponti su cui sono stati sviluppati gli esperimenti con le formiche.*

Un secondo esperimento con un ponte formato da rami di differente lunghezza ha introdotto un nuovo elemento. Anche se si ha eguale probabilità di imboccare uno dei due rami del ponte, le formiche che scelgono il ramo più corto troveranno prima il cibo. Così, le scie di feromone copriranno tale ramo che avrà maggiore probabilità di essere percorso.

DALLA BIOLOGIA ALL'OTTIMIZZAZIONE

Realizzare un modello matematico (algoritmico) ispirato al modo di agire delle formiche significa costruire un certo numero di formiche artificiali che grazie alla cooperazione appena descritta, ossia attraverso lo scambio di informazioni, concorrono alla risoluzione di un problema. Su un grafo adottato come struttura di base per il nostro problema, si introducono delle formiche. L'idea è imitarle. Tra le prime applicazioni di ACO si scorge la soluzione del conosciuto problema del commesso viaggiatore (TSP). Successivamente importanti risultati sono stati ottenuti in svariati campi, dagli algoritmi genetici al "network routing", ed in altri ancora. Per avere una prima idea vediamo come si può applicare ACO a TSP. Ricordo che si tratta del problema di un commesso viaggiatore il quale per lavoro deve visitare n città la cui distanza (tra ogni coppia) è nota, e che ha il vincolo di passare una sola volta per ogni città; il suo obiettivo è trovare il tragitto più corto. Si tratta quindi di individuare un ciclo Hamiltoniano su di un grafo corrispondente. Con ACO il problema si affronta predisponendo un certo numero di formiche artificiali che si muovono sui vertici del grafo. Bisognerà concepire una variabile feromone associata agli archi e che sarà di volta in volta modificata da passaggio delle formiche. Ad ogni passo dell'algoritmo una formica che si trova nel nodo i, tra i possibili nodi che può raggiungere (i vincoli del problema impongono di scartare quelli già visitati) sceglierà secondo un processo stocastico un nodo j con una probabilità proporzionale alla quantità di feromone presente sull'arco.

FEROMONE

I feromoni (dal greco pherein "trasportare" e hormon "eccitare") sono sostanze chimiche, che producono segnali attivi a basse concentrazioni, prodotte ed escrete da un individuo, animali in genere. Sono in grado di suscitare delle reazioni specifiche di tipo fisiologico e/o comportamentale in altri individui della stessa specie che vengono a contatto con esse.
Una prima classificazione avviene in funzione dell'effetto prodotto:

- **feromoni traccianti (trace) che rilasciati da un individuo vengono seguiti da appartenenti alla stessa specie come una traccia. Si tratta della categoria su cui si basa ACO.**
- **feromoni di allarme (alarm) che vengono emessi in situazioni di pericolo, inducendo un maggiore stato di vigilanza in quanti li captano.**
- **feromoni innescanti o scatenanti (primer) che inducono nel ricevente modificazioni comportamentali e/o fisiologiche a lungo termine.**
- **feromoni liberatori o di segnalazione (releaser) che scatenano comportamenti di aggressione o di accoppiamento nell'animale che li capta.**
- **feromoni sessuali (sex) che producono attrattiva in altri individui.**

Ad esempio le api regine rilasciano un tipo di feromone che inibisce l'attività riproduttiva della operaie.

UNA PROBLEMA DI OTTIMIZZAZIONE COMBINATORIA

Nel presentare le formulazioni dei modelli a seguire cercherò di attenermi al simbolismo usato da Marco Dorigo e dagli altri ricercatori di ACO, cosicché chi volesse approfondire (si vedano i box relativi) non si troverà disorientato rispetto a questa presentazione. Cominceremo con una formulazione generale per un problema di ottimizzazione combinatoria. È conosciuta anche come metaeuristica, proprio perché si comporta come un solutore di carattere generale, da adottare per un intero pacchetto di differenti problemi. Il modello P è caratterizzato dalla terna: <S, W, f> dove:

- Lo spazio di ricerca S è definito su un insieme finito di variabili di decisione discrete Xi, con i compreso tra 1 e n;
- Un insieme W di vincoli sulle variabili;
- Una funzione obiettivo f restituisce un valore reale a fronte di un valore appartenente a S, essa deve essere minimizzata.

Con vij indichiamo un'istanza della variabile decisionale Xi . Le variabili vij saranno definite in rispettivi domini. Una soluzione s appartenente a S sarà ottima se non esiste nessun altra soluzione che applicata a f sia minore di f(s).
Nella costruzione del modello si deve tenere conto di ogni possibile soluzione componente a cui è associato un valore di feromone. Il valore di feromone è indicato con il valore ti,j e la soluzione componente corrispondente è indicata con ci,j . L'insieme di tutte le soluzioni è C (ben rappresentata con una matrice).
Ecco come può essere descritto per macropassi l'algoritmo di metaeuristica.

```
Configura parametri e inizializza livelli di feromone
While non è verificata la condizione di terminazione do
        AntSolution
        RicercaLocale
    AggiornaFeromone
End while
```

La prima delle fasi indicata con AntSolution costruisce un insieme di m formiche artificiali e configura una prima soluzione componente $c_{i,j}$. Inoltre, pone a zero la prima delle soluzioni parziali $_{s}p=0$. Ad ogni passo dell'algoritmo la soluzione parziale si aggiorna di nuove possibili soluzioni che vengono ottenute applicando N($_{s}$p) che sarà un sotto insieme dello spazio delle soluzioni C, ovviamente, nel rispetto dei vincoli W. La scelta della soluzione componente da N($_{s}$p) è un processo stocastico che tiene conto dei livelli di feromone. Con *RicercaLocale* si occupa di migliorare localmente la soluzione prima di aggiornare i livelli di feromone. Tale fase è molto delicata ed è oggetto di studio al fine di ottenere risultati soddisfacenti. La fase *AggiornaFeromone* tiene conto della presenza delle formiche su alcune componenti allo scopo di ottenere una migliore e più rapida convergenza verso la soluzione ottima.

USANDO GRAFI

L'algoritmo appena formulato indica la strada maestra da seguire per realizzare ACO. Esso è più facilmente comprensibile se anziché ad un problema di ottimizzazione combinatoria, che comunque rimane il più generale, si ci riferisce ad problema di attraversamento di grafi. Noteremo che gli oggetti appena indicati assumono un significato di facile riscontro. Si considera un grafo Gc(V,E) costituito da due insiemi: di nodi (o vertici) V e di archi E. Il grafico lo si può associare al set di soluzioni componenti C. In particolare, le formiche si muovono sui nodi che per il problema del commesso viaggiatore rappresentano le città. Gli insetti nello spostarsi aggiornano di continuo la soluzione parziale. Inoltre, modificano le quantità di feromone negli archi che percorreranno. Le soluzioni componente sono rappresentate da $c_{i,j}$, nel caso specifico ciò indica che la città j deve essere visitata dopo i. Con i valori $t_{i,j}$ indicheremo la nuova quantità di feromone che si ha sull'arco (i,j) dopo il passaggio della formica.

L'ALGORITMO ANT SYSTEM

Esistono diversi algoritmi ACO, il più remoto e significativo è stato introdotto da Marco Dorigo ed è conosciuto come *ant system*. Si tratta di un sistema discreto che avanza ad ogni iterazione. Tutti i livelli di feromone vengono aggiornati in base all'azione delle m formiche artificiali. La quantità $t_{i,j}$ indicante il feromone presente sull'arco (i,j) viene aggiornata secondo questa espressione.

$$\tau_{i,j} = (1-\rho)\cdot\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau^{k}_{i,j}$$

Dove *m* è il numero di formiche. r è coefficiente di evaporazione, specifica con il passare delle iterazioni quanto feromone viene liberato; è un valore compreso tra 0 e 1 ed è facile comprendere che valori vicini all'uno indichino elevati gradi di evaporazione, in pratica dopo una iterazione gran parte del feromone si volatilizza. Valori vicini allo zero al contrario modellano una lenta evaporazione. La seconda parte dell'espressione è il contributo di feromone dovuto dalle m formiche nell'ultima iterazione. È chiaro che "l'attrattiva", ossia la quantità di feromone, aumenta se il contributo delle singole formiche supera la quantità evaporata, e ciò ad intuito si ottiene quando quel ramo viene percorso da un "certo" numero di formiche. Ovviamente, non tutte le formiche che percorrono un arco forniscono lo stesso contributo e va da se che il mancato passaggio per l'arco dà contributo nullo. La quota aggiunta dalla formichina k che è passata dall'arco (i,j) è stata indicata come elemento della sommatoria con D, questo valore è pari a Q/L_k. Dove Q è una costante e L è la lunghezza del tour costituito dalla formica k. A questo punto ci chiediamo come le formiche scelgano gli archi da percorre tra quelli permessi (che non siano già stati visitati). Come descritto si tratta di un processo stocastico che dipende dalla quantità di feromone che ciascun arco presenta. Più precisamente la probabilità che la formica k dalla città i scelga di raggiungere la città j può essere espressa come segue:

$$p^{k}_{ij} = \frac{\tau^{\alpha}_{ij}\cdot\eta^{\beta}_{ij}}{\sum_{l\in N(s^{p})}\tau^{\alpha}_{ij}\cdot\eta^{\beta}_{ij}}$$

Questo se la soluzione componente ci,j appartiene a N(sp), altrimenti la probabilità è nulla. I due parame-

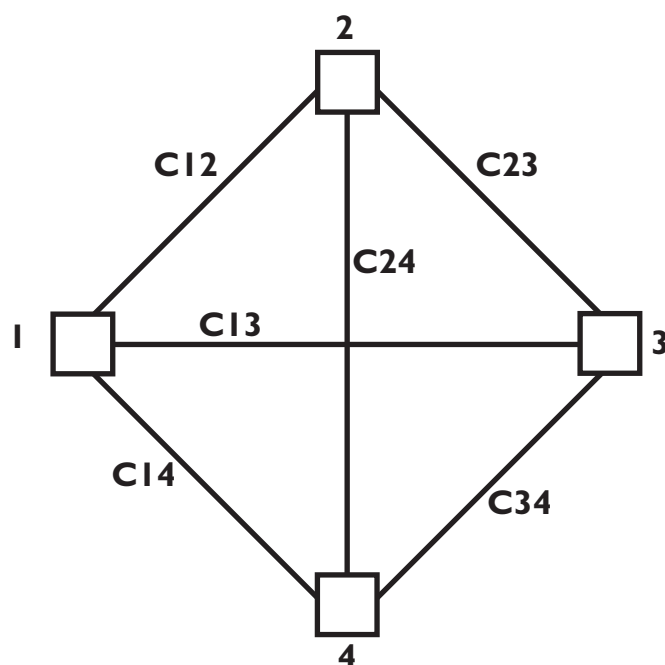



Fig. 2: "Esempio di grafo su cui applicare ACO"

tri a e b sono delle costanti che esprimono il peso da assegnare al feromone in funzione della variabile h. Questa ultima variabile è l'inverso della distanza tra i nodi interessati. Dall'analisi dell'espressione che è un rapporto emerge evidente la dipendenza della probabilità dai livelli di feromone e dalla distanza, legati in un unico fattore (numeratore). La sommatoria al denominatore indica il totale di tali fattori a partire dalla città i. Risulta che la probabilità maggiore si avrà per quel nodo che esprime il maggiore fattore, che in definitiva è il rapporto tra livello di feromone e distanza modificato dall'elevazioni alle costanti prima descritte. Riesaminando l'algoritmo generale prima descritto alla luce degli strumenti matematici appena menzionati si scopre che l'espressione per il calcolo di t (livello di feromone) si può usare all'interno della procedura AggiornaFeromone, mentre RicercaLocale utilizzerà la probabilità calcolata per esplorare nuove soluzioni componenti. La prima procedure per la configurazione si occuperà di dimensionare tutti i parametri incontrati, che ricordo sono r, a, b e Q. Tali parametri dipendono dal problema specifico che si intende risolvere.

ALGORITMO MMAS

Molti altri sviluppi si sono registrati dalla prima versione di Ant system appena presentata. Tra tutte ne presentiamo uno: Max-Min Ant system (conosciuto con la sigla MMAS) , per gli altri rimando ai riferimenti sul web e alla bibliografia. In questa nuova versione sono presenti dei cambiamenti sulla modellazione del feromone a cui sono imposti dei limiti minimi e massimi e il cui aggiornamento è demandato ad una sola formica che si distingue per performance. La nuova espressione per il calcolo del feromone è adesso descritta così:

$$\tau_{i,j} = [(1-\rho)\cdot\tau_{i,j} + \Delta\tau^{best}{}_{i,j}]^{\tau\,max}_{\tau\,min}$$

I livelli di feromone sono quindi compresi nell'intervallo descritto. La migliore variazione (delta) di feromone indicata con best non è altro che il rapporto 1/Lbest, con L lunghezza del percorso della migliore formica, sempre se (i,j) appartiene al miglior tour; è infatti 0 altrimenti. I bound sono come per l'algoritmo precedente strettamente dipendenti dal problema che si sta affrontando e quindi vanno calibrati rispetto ad essi. Va evidenziato che sebbene per alcune classi di problemi l'algoritmo MMAS abbia dato migliori risultati di Ant System, il secondo è quello che più si avvicina al reale comportamento della comunità di formiche, proprio perché le quantità di feromone che ad ogni iterazione si aggiornano sono il risultato del contributo di diverse formiche, proprio come accade in natura. Il metodo MMAS ci ricorda, invece, gli algoritmi evolutivi che abbiamo esaminato negli scorsi appuntamenti, anche se l'individuazione di una formica con le migliori performance non è da ritenersi un processo di selezione. Piuttosto ritorna intrinsecamente l'indice di valutazione di fitness.

CAMPI DI APPLICAZIONE

L'algoritmo inizialmente costruito sul problema del commesso viaggiatore, successivamente è stato testato su molti altri problemi NP-Hard. Le categorie di problemi su cui è stato applicato ACO sono:

- Routing: ad esempio per l'assegnazione di buoni o problemi sulle reti;
- assegnamento: il classico problema in cui bisogna far corrispondere entità a risorse (come il caso di oggetti a luoghi) ottimizzando una funzione obiettivo e rispettando determinati vincoli;
- scheduling in cui bisogna gestire risorse e compiti in un fissato tempo;
- Sottoinsiemi: come il caso di Knapsack.

Per queste categorie la metaeuristica ha fornito risultati soddisfacenti anche se si è rilevato delicato e a volte ostico il lavoro di settaggio dei parametri di progetto che differiscono a seconda del campo di applicazione dell'algoritmo. Vanno ricordati i successi nel campo del routing su reti che sono stati raggiunti grazie al contributo di molti studiosi di tutto il mondo. Significativi i risultati anche nel campo industriale dove la ricerca si è concretizzata in effettivi utilizzi di automazione industriale e di ottimizzazione di percorsi. Emblematico è il caso della società di logistica svizzera AntOptima che ha sviluppato una serie di tool che si basano proprio su ACO. Tra gli strumenti vi è *DYVOIL* che gestisce la distribuzione di carburante per riscaldamento su mezzi di diverse dimensioni usato per la prima volta dalla società svizzera Pina Petroli.

CONCLUSIONI

È davvero sorprendente verificare i brillanti risultati ottenuti con ACO. Non ripeterò i campi in cui si stanno sviluppando significative conquiste, mi interessa qui sottolineare il vivo interesse mostrato per la questione da studiosi e programmatori che hanno pienamente approvato lo studio introdotto da Marco Dorigo. La Swarm Intelligence è un campo in continua evoluzione che mostra come millenni di adattamento applicati all anatura possano costituire un modello valido da replicare in programmazione

Fabio Grimaldi